An ML-Aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers

Dinesh Cyril Selvaraj, Shailesh Hegde, Nicola Amati, Francesco Deflorio, and Carla Fabiana Chiasserini, *Fellow, IEEE*

***Abstract*—The richness of information generated by today's vehicles fosters the development of data-driven decision-making models, with the additional capability to account for the context in which vehicles operate. In this work, we focus on Adaptive Cruise Control (ACC) in the case of such challenging vehicle maneuvers as cut-in and cut-out, and leverages Deep Reinforcement Learning (DRL) and vehicle connectivity to develop a data-driven cooperative ACC application. Our DRL framework accounts for all the relevant factors, namely, passengers' safety and comfort as well as efficient road capacity usage, and it properly weights them through a two-layer learning approach. We evaluate and compare the performance of the proposed scheme against existing alternatives through the CoMoVe framework, which realistically represents vehicle dynamics, communication and traffic. The results, obtained in different real-world scenarios, show that our solution provides excellent vehicle stability, passengers' comfort, and traffic efficiency, and highlight the crucial role that vehicle connectivity can play in ACC. Notably, our DRL scheme improves the road usage efficiency by being inside the desired range of headway in cut-out and cut-in scenarios for 69% and 78% (resp.) of the time, whereas alternatives respect the desired range only for 15% and 45% (resp.) of the time. We also validate the proposed solution through a *hardware-in-the-loop implementation*, and demonstrate that it achieves similar performance to that obtained through the CoMoVe framework.**

***Index Terms*—Machine learning-based vehicle applications, connected vehicles, vehicle dynamics, adaptive cruise control.**

I. Introduction

RECENT report by the World Health Organization (WHO) indicates that nearly 1.35 million people die in road accidents, and approximately 20–50 million people suffer non-fatal injuries yearly. Also, traffic congestion takes a substantial toll on public health and economy because of the polluted air, people's commuting time, and fuel consumption [1], [2]. In this context, Connected Autonomous Vehicles (CAV) can play an essential role, as they can mitigate traffic externalities, especially safety and traffic efficiency. Both vehicles and road infrastructures are increasingly equipped with sensing computational equipment to assist the driver, as well as with vehicle-to-everything (V2X) communication devices to facilitate data exchange. As a result, a CAV can gather an enormous amount of data promoting the development of Machine Learning (ML) models to further improve passengers' safety and comfort.

Among the Advanced Driver Assistance Systems, the Adaptive Cruise Control (ACC) is one of the most popular applications in new vehicles generations, and it seemingly performs well under most car-following scenarios. However, there are few challenging scenarios where the human has to be alert and take control of the vehicle to perform a safe maneuver over the ACC [3]. One such scenario is given by the lane change maneuvers which are more common on the roads, and responsible for 7.6% of the car crashes in the US [4]. To overcome such limitations of the traditional ACC, we propose an ML-based ACC application that leverages the information collected through both sensors and communication devices. Such application can improve not only safety but also comfort and traffic efficiency since it substantially reduces the traffic shock waves that usually occur during challenging maneuvers.

More specifically, the framework we propose, called 2-Layer Learning Cooperative ACC (2LL-CACC), accounts for CAVs' road efficiency, safety, and comfort, as follows. Efficiency is measured by the headway metric – a proxy way to measure the inter-vehicle distance in a traffic stream [5], [6], [7]. Safety is expressed in terms of the longitudinal slip ratio and the Time-To-Collision (TTC), where the former is the amount of slip experienced by pneumatic tires on the road surface, while the latter represents the time it takes for two vehicles to collide. Finally, comfort is measured through the jerk metric, defined as a rate of change in the vehicle's acceleration.

As sketched in Fig. 1, 2LL-CACC aims at finding the best tradeoff among road efficiency, safety, and comfort by using a Deep Reinforcement Learning (DRL) where the reward function is an ML-driven weighted sum of the three metrics. The top layer hosts a Random Forest Classifier [8] to assess the current contextual information, while the lower one includes a Deep Deterministic Policy Gradient (DDPG) [9] algorithm that aims to maximize the cumulative reward by mapping the states and action through an optimal policy. Thanks to such a 2-layer ML-based approach, 2LL-CACC can adapt to the operational context and effectively selects the acceleration to adopt, thus overcoming the limitations of the traditional ACC in coping with

Manuscript received 22 August 2022; revised 25 October 2022 and 16 November 2022; accepted 20 November 2022. Date of publication 24 November 2022; date of current version 20 March 2023. This work was supported in part by NPRP-S 13th Cycle under Grant NPRP13S-0205-200265, in part by Qatar National Research Fund (A Member of Qatar Foundation), and in part by European Union Through the CONNECT Project under Grant 101069688. *(Corresponding author: Carla Fabiana Chiasserini.)*

The authors are with the CARS@Polito, Politecnico di Torino, 10129 Torino, Italy (e-mail: dinesh.selvaraj@polito.it; shailesh.hegde@polito.it; nicola.amati@polito.it; francesco.deflorio@polito.it; carla.chiasserini@polito.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2022.3224656.

Digital Object Identifier 10.1109/TIV.2022.3224656

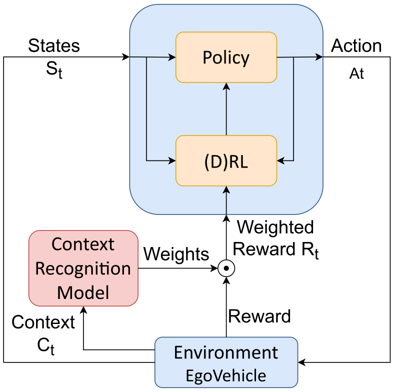

Fig. 1. Architecture of 2LL-CACC scheme.

challenging traffic situations. To demonstrate this, we primarily address road traffic scenarios where the ego vehicle follows a short-distance/low-velocity lead vehicle, as it typically occurs during cut-in and cut-out scenarios.

Our main contributions can thus be summarized as follows:

(i) We present the 2LL-CACC, an ML-aided DRL framework that employs a two-layered learning strategy to accomplish road efficiency, safety, and comfort objectives. The two layers host the Context Recognition Model and the DRL model, respectively. The role of the Context Recognition Model is to recognize the current contextual information and appropriately weigh the reward components, i.e., road efficiency, safety, and comfort. Subsequently, the weighted reward components assist the DRL model convergence by providing valuable feedback on the DRL learning process.

(ii) To achieve the above objectives and adequately represent the environment, the DRL states exploit information about the lead vehicle and its relation to the ego vehicle in terms of its lead vehicle's acceleration, headway, and relative velocity. Furthermore, vehicle stability-related states, such as longitudinal slip and road friction coefficient, are used to evaluate the vehicle's stability. As rewards, we use headway as a traffic efficiency indicator, jerk to assess comfort, and slip to ensure vehicle stability. The reward components are modeled to provide positive/negative reinforcement to the agent as feedback.

(iii) Specifically, for the aggressive driving scenarios, we have introduced a V2X-supported gradual-switching technique that facilitates the ego vehicle to change focus on the lane-changing vehicle safely and steadily. Unlike the car-following scenario, gradual switching is crucial for the early identification of lane-changing vehicles and smooth transition between the vehicles to prevent the deterioration of the target key performance indicators.

(iv) We present a detailed process flow of the Hardware-In-the-Loop (HIL) implementation that facilitates the real-time deployment of the PyTorch-based DRL agent in the dSPACE SCALEXIO AutoBox through the MathWorks environment. Also, the HIL validation demonstrates that 2LL-CACC can be actually implemented in a real-world vehicle and that it achieves a similar outcome as in the CoMoVe simulations.

Overall, the proposed system uses a content recognition model to assess the contextual information, the DRL model to drive the ego vehicle in an efficient, safe, comfortable way, and finally, the gradual switching to identify the lane-changing vehicles and adequately manipulate the DRL states to take suitable decisions.

The rest of the paper is organized as follows: Section II discusses relevant previous work and highlights our novel contributions. Section III describes the 2LL-CACC framework and explains how V2X communication is exploited, while Section IV and Section V detail, respectively, the integration with the CoMoVe framework and the process flow of the HIL implementation. Section VI presents the performance of 2LL-CACC against state-of-the-art alternatives. Finally, Section VII draws our conclusions and discusses future work.

II. Related Work

The Adaptive Cruise Control application efficiently controls the longitudinal speed of the vehicle for simple car-following scenarios, while such complex conditions like cut-in or cut-out maneuvers can be highly challenging [3], [10], as the inter-vehicle distance may change dramatically. In particular, a defensive response to the cut-in/cut-out vehicles may greatly affect traffic efficiency [11], while an overly aggressive reaction leads to collision with very high probability [4]. It is thus critical that automated/autonomous vehicles overcome the current limitations to ensure safety. To assist vehicles in such complex situations, ML techniques are widely adopted. In particular, (D)RL algorithms have been preferred to other ML approaches, since they effectively deal with uncertain and partially observable environments [12]. Several works [13], [14], [15], [16], [17] have explored the usage of (D)RL-based algorithms to improve vehicle performance in complex scenarios. In particular, [13] leverages a DRL-based CACC algorithm that exploits information from vehicle's RADAR and vehicle-to-vehicle (V2V) communication to maintain the desired headway with the lead vehicle. Even though V2V communication can help to identify lane-changing scenarios beforehand, [13] only focuses on optimizing headway, thus overlooking the passenger comfort or vehicle stability. The traditional ACC also suffers from similar inadequacies, as it does not consider the environmental factors while controlling the longitudinal vehicle movements. The DRL-based framework in [15] addresses some drawbacks of [13], by using a multi-objective reward function to optimize vehicle's safety, comfort and efficiency. It also considers a continuous action space, unlike the DRL framework in [13] which can only select an action from a pre-defined discrete action space. However, [15] only considers a linear model to simulate the vehicle behavior, which is often not suitable to represent a vehicle in real-world conditions. Furthermore, prior art has not considered vehicle stability under different road conditions as an objective, which is an integral part of the passenger's safety. To address this limitation, our framework utilizes longitudinal slip ratio and road friction coefficient to ensure the vehicle's stability. Even though we obtain these parameters from the simulation models, one can estimate them in real-life situations by leveraging the estimation techniques proposed in, e.g., [18], [19], [20].

Looking at the cut-in scenario, [16] presents a DRL framework tailored to deal with cut-in events and car-following scenarios. [16] uses a two-step process: (i) a deep neural network trained to predict the cut-in maneuver, and (ii) a Double Deep Q Network (DDQN) to train the DRL model for the cut-in scenario. As part of the second step, the authors develop an Experience Screening, a pre-training process where multiple DRL simulations are performed for a set of pre-defined scenarios, and the best experiences (states, actions, rewards, transition states) of each scenario are stored in an experience pool. Later, the DDQN samples the data from the experience pool for faster training convergence and generalization across different scenarios. We take this study as one of the benchmarks against which we compare our scheme. However, since the dataset used in [16] is not publicly available, we had to compared our framework to the vanilla DDQN algorithm, which is the core component of the algorithm proposed in [16]. It is also worth stressing that, differently from the framework we propose, the DDQN algorithm only supports a discrete action space. Thus, the degree of freedom to choose an appropriate action is limited, compared to 2LL-CACC.

As for non-ML-based methods, [21], [22], [23] focus on improving the traditional ACC to handle the cut-in vehicles. [22] presents a (C)ACC algorithm for platooning in vehicle cut-in/cut-out situations. It uses a Proportional-Derivative (PD) controller, which takes relative velocity and distance between the lead and ego vehicle as input and outputs the desired ego vehicle acceleration. Nevertheless, in our conference paper [24], we compared the performance of a similar (C)ACC controller to our proposed DRL framework, and the results showed that the DRL framework can achieve better results than the approach in [22]. [23] proposes instead Model Predictive Control (MPC) for cut-in maneuvers with safety and comfort objectives. However, [25] shows that MPC suffers high computation and time complexity, while a model-free DRL model can be trained offline and provide results promptly.

In a (D)RL framework, the representation of a reward function is critical, as it quantifies the value associated with each state and action pair and assists the agent in learning an optimal policy. [26] remarks that (D)RL with sparse rewards can lead to instability and suboptimal policy convergence. Likewise, each reward component should be weighted optimally in a multi-objective DRL agent to achieve the desired outcome and faster convergence. Few studies [27], [28] use supervised reward shaping techniques to assist the sparse rewards setup, which is a different approach from ours, as we focus on predicting optimal weights for each reward component according to the current contextual information.

In general, (D)RL frameworks employ recognized simulators to validate their agent's performance [29]. In our work, we employ the CoMoVe simulation framework [30]: a sophisticated validation tool that can realistically simulate both detailed vehicle dynamics and communication models.

Finally, we mention that a preliminary version of this work has been presented in our paper [24]. With respect to [24], (i) we now develop a two-layer ML-based approach, with an ML classifier dynamically determining the setting of the weights for the three reward components in the DRL agent; (ii) the DRL framework is enhanced to identify lane-changing scenarios in advance and actuate gradual-switching strategy to the new lead vehicle assuring comfort, safety, and efficiency, and (iii) the HIL implementation demonstrates the deployability of 2LL-CACC in actual vehicles.

III. The 2LL-CACC Framework

In this section, we describe the 2LL-CACC scheme, which aims to learn an optimal decision-making strategy for the ego vehicle, ensuring an efficient, safe, and comfortable driving experience. As depicted in Fig. 1, 2LL-CACC comprises two layers: the top one hosts an ML model to access the current context and scenario characteristics; the lower one focuses on the DRL agent attributes to learn an optimal policy.

At any given time t, the ego vehicle traveling through a road traffic scenario provides information about the *environment*, specifically, neighboring vehicles and road conditions, to the DRL agent and Context Recognition model as state $s(t) \in \mathcal{S}$ and context $c(t) \in \mathcal{C}$ (resp.). Given $s(t)$, the role of the DRL framework is to attain efficient, safe, and comfortable driving 2LL-CACC by maintaining optimal speed, according to headway, slip, and jerk values through the agent's decision-making policy. Based on the input state, $s(t) \in \mathcal{S}$, the DRL agent takes action ($\mathcal{A}$) to change the behavior of the ego vehicle by either accelerating or decelerating it. As a response to the action, the agent gets a reward from the environment. The representation of states, actions, and reward in the DRL framework assists the agent in learning the optimal policy.

In our study, the reward comprises three components, namely, headway, slip, and jerk, to model efficient, safe, and comfortable driving. However, equally weighted reward components may not provide optimal feedback to the DRL agent, as, depending on the situation experienced by the ego vehicle, a component may be more important and hence should be weighted more. Examples include the case where road pavement conditions are particularly slippery and vehicle stability has to be ensured with highest priority, or the case where the ego vehicle has high driving speed and should maintain a sufficient headway. Thus, each reward component should be weighted depending upon the current context, as the latter impacts the learning process directly. To do so, it is necessary to derive the relation between features that impact the reward components and the corresponding weights. To this end, we introduce the Context Recognition Model, which leverages a Random Forest Classifier to infer such a relationship and determine the weight to be associated with each reward component based on the current context $c(t)$. Subsequently, the predicted weights are used to regulate their corresponding reward components, and the sum of the weighted rewards facilitates the DRL model in learning an optimal policy. Fig. 2 depicts an overview of the proposed 2LL-CACC framework.

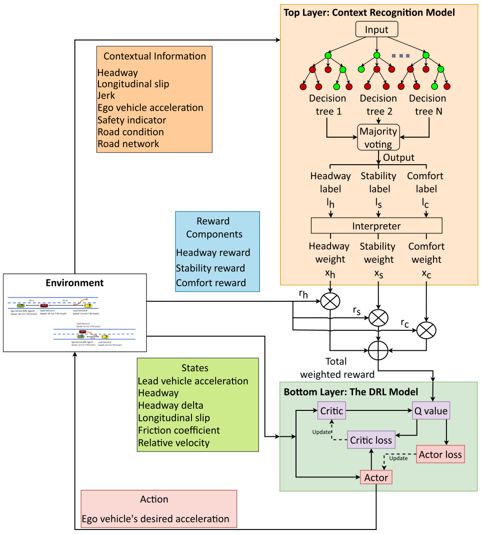

Fig. 2. An overview of the proposed 2LL-CACC framework.

TABLE I
NOTATIONS

Symbols	Description
$\mathcal{C}$	Contextual Information
$\mathcal{S}$	DRL State Space
$\mathcal{A}$	DRL Action Space
$\mathcal{Z}$	Experience Replay Buffer
x_h	Headway Reward Weight Coefficient
r_h	Headway Reward Component
x_s	Stability Reward Weight Coefficient
r_s	Stability Reward Component
x_c	Comfort Reward Weight Coefficient
r_c	Comfort Reward Component
α	Lead Vehicle Acceleration
ϑ	Headway
$\Delta\vartheta$	Headway Derivative
ξ	Longitudinal Slip
μ	Road Friction Coefficient
ν	Relative Velocity
j	Jerk
$\ddot{x}$	Ego Vehicle's Acceleration
χ	Safety Indicator, corresponding to the Time-to-Collision (TTC) value
ψ	Road Condition, based on the road friction coefficient (μ)
ω	Road Network, describes the current road traffic scenario
$\mathcal{Q}(s, a\|\beta)$	Parameterized State-Action value function with β as parameters
$\pi(s\|\eta)$	Parameterized Policy function with η as parameters

In the following, Section III-A details the top layer hosting the Context Recognition model, while Section III-B presents the lower layer hosting the DRL framework. The notations used in Section III-A and Section III-B are summarized in Table I.

A. Top Layer: Context Recognition Model

As mentioned above, we use an ML model to predict the weights of each reward component, based on the current situation. Specifically, we uses a Random Forest Classifier (RFC) [8], [31] and train it to interpret the current contextual information through selected input features ($\mathcal{C}$), and output the optimal weight class for the headway (l_h), stability (l_s), and comfort (l_c) reward components.

1) Preliminaries: In general, RFC is a powerful ensemble algorithm that has been proved to handle high dimensionality problems efficiently. It suits the problem at hand particularly well, since we have a set of input features that must be mapped into three weight labels accordingly. In a nutshell, the RFC builds a set of independent decision trees and aggregates them together to get accurate predictions. Notably, the Random Forest model utilizes the bootstrap aggregation method to mitigate the high variance issue observed in single decision tree techniques. With the help of bootstrap aggregation, each decision tree samples a random subset of data from the original dataset and it uses a random subset of input features to build the tree. Because of the randomness, the decision trees are less correlated and produce better prediction outputs than a single decision tree. Essentially, a decision tree aims to split the data into homogeneous branches to determine the outcome. The tree includes two types of nodes (i) a decision splitting the data into two subsets (branches), and (ii) a leaf node representing an outcome decision. With the help of the Gini Index (GI) [31], each decision node determines a splitting criterion based on a specific feature and a threshold, which results in fewer samples of heterogeneous classes in each subset $\mathcal{H}$. At each subset, the GI is calculated as:

$$GI(\mathcal{H}) = 1 - \sum_{i=1}^{n} p_i^2 \tag{1}$$

where n is the total number of classes, and p_i the number of samples in subset $\mathcal{H}$ that belong to the $i-$th class normalized to $|\mathcal{H}|$'s' cardinality. Then, the weighted average of each subset's GI is used to identify the best criteria to split the data. Given subsets $\mathcal{H}_1$ and $\mathcal{H}_2$, the weighted GI is given by:

$$GI_w(\mathcal{H}_1, \mathcal{H}_2) = \frac{n_1}{n} GI(\mathcal{H}_1) + \frac{n_2}{n} GI(\mathcal{H}_2) \tag{2}$$

where n_1 and n_2, represent the number of samples in subsets $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, and n is the total number of rows in $\mathcal{H}_1 \cup \mathcal{H}_2$. Similarly, GI_w is calculated for different splitting criteria, and the criterion with minimum GI_w is used to split the data. Indeed, the smaller GI value means better splitting criteria with a higher percentage of homogeneous classes in the subsets. The decision node continues to split the data till all values in each subset are homogeneous, i.e., belong to the same class. However, the splitting is controlled by the maximum tree depth parameter to tackle overfitting. In the decision-making (i.e., inference) phase, the input data traverse the decision tree from decision nodes to the leaf node, where the majority class in the leaf node is predicted as the output label. The output label is decided based on the majority vote of all decision trees.

2) Context Recognition Model: The context recognition model employs the Random Forest Classifier to predict the weights of the reward components based on the current contextual information. Therefore, the RFC takes input features that concern the ego vehicles' objectives to choose the corresponding labels for the reward components. We have three reward components representing headway (r_h), stability (r_s), and comfort (r_c).

To support the classification model, we discretize the weight values into 20 bins and numerically labeled each bin (e.g., label 1 represents [0.0, 0.05], label 2 [0.05, 0.1], etc.). We chose classification rather than regression algorithms because the predicted values vary considerably in regression, causing the DRL model to map a similar state-action pair with different rewards, and, hence, slowing down convergence. At a given time t, the input features headway ($\vartheta(t)$), jerk ($j(t)$), and longitudinal slip ($\xi(t)$) play a crucial role, as they represent the three main objectives of the ego vehicle. They are formulated as:

$$\vartheta(t) = \frac{\Delta P_{lead}(t)}{\mathcal{V}_{ego}(t)} \tag{3}$$

$$j(t) = \frac{\ddot{x}(t) - \ddot{x}(t-1)}{\tau}, \tag{4}$$

$$\xi(t) = \begin{cases} \frac{\mathcal{V}^R_{ego}(t) - \mathcal{V}^W_{ego}(t)}{\mathcal{V}^R_{ego}(t)}, \ \ddot{x}(t) >= 0 \\ \frac{\mathcal{V}^R_{ego}(t) - \mathcal{V}^W_{ego}(t)}{\mathcal{V}^W_{ego}(t)}, \ \text{otherwise} \end{cases} \tag{5}$$

where $\Delta P_{lead}(t)$ is the relative distance between lead and ego vehicle, $\mathcal{V}_{ego}(t)$ is the ego vehicle's velocity, $\mathcal{V}^R_{ego}(t)$ is the ego vehicle's tire tangential velocity, $\mathcal{V}^W_{ego}(t)$ is the ego vehicle's wheel ground point velocity, $\ddot{x}(t)$ is the ego vehicle's acceleration at time t and τ is the sampling interval of the framework. Apart from them, the ego vehicle acceleration ($\ddot{x}(t)$) at time t helps identify the vehicle's current operating range (braking or speeding up), which affects the objectives.

Likewise, a safety indicator is used to determine critical situations and further discount the comfort factor in case of imminent danger. In this work, we use Time-To-Collision (TTC) to identify potential collision situations, representing the time it takes for two vehicles to collide, if their speed is not modified. The TTC at time t is formulated as:

$$TTC(t) = \frac{\Delta P_{lead}(t)}{\nu(t)} \tag{6}$$

where $\nu(t)$ represents the relative velocity between the lead and ego vehicles at time t. Therefore, the safety indicator at time t ($\chi(t)$) is computed as:

$$\chi(t) = \begin{cases} 1, \ TTC(t) > 4\,s \\ 0, \ TTC(t) \leq 4\,s \end{cases} \tag{7}$$

where the 4-s threshold is set based on [32].

Furthermore, the road friction coefficient ($\mu(t)$) at time t is directly related to vehicle stability, where an abrupt acceleration change often leads to instability in low-friction roads. The road condition ($\psi(t)$) is defined as:

$$\psi(t) = \begin{cases} 1, \ 0.7 \leq \mu(t) \leq 1 \\ 2, \ 0.4 \leq \mu(t) < 0.7 \\ 3, \ \mu(t) < 0.4\,. \end{cases} \tag{8}$$

Finally, ($\omega(t)$) represents the road traffic scenario at time t, as the ego vehicle' behavior may significantly vary, e.g., from urban intersections to highway scenarios. In addition, the road traffic scenarios are expressed as unique discrete values to benefit the learning process.

To summarise, at the generic time-step t, context ($c(t)$) is represented by the following features:

- headway ($\vartheta(t)$) representing the distance between ego and lead vehicle;
- longitudinal slip ($\xi(t)$) representing the ego vehicle's stability;
- jerk ($j(t)$), i.e., the comfort factor;
- ego vehicle's acceleration ($\ddot{x}(t)$);
- $\chi(t)$, a binary value that indicates whether the TTC drops below a fixed safety threshold or not;
- road condition ($\psi(t)$) describing the road friction coefficient ($\mu(t)$) in the form of slippery, wet, or dry conditions;
- road network ($\omega(t)$) representing the current road traffic scenario.

At every time-step, the RFC model takes the current context ($c(t)$) as input and predicts the optimal weight label for headway, stability, and comfort reward components. Then, labels (l_h, l_s, l_c) are converted into values (x_h, x_s, x_c) according to their discretized bins. For example, with reference to the above example about the bins' labels, if the RFC model predicts 1 as headway label l_h, the corresponding headway weight is a random uniform value between 0 to 0.05.

B. Bottom Layer: The DRL Model

We now provide a detailed description of the DRL model, starting with some preliminaries on DRL and then introducing the solution we designed.

1) Preliminaries: The goal of a RL model is to learn an optimal decision-making strategy by repeatedly interacting with an environment that provides positive or negative feedback as a reward for the current behavior. Its main components are: state-space ($\mathcal{S}$), i.e., a representation of the environment; action space ($\mathcal{A}$), a set of actions an agent can take to interact with the environment; rewards ($\mathcal{R}$), numerical feedback from the environment; policy ($\pi(s)$), a decision-making strategy that characterizes the mapping from states to actions; value function ($\mathcal{Q}_\pi(s, a)$), which indicates the expected future return from the state-action pair. The policy and value function facilitate the agent to take a sequence of actions that maximizes the cumulative discounted reward received from the environment.

The RL problem is generally modeled as a Markov Decision Process (MDP). At any given time step t, the MDP is represented through a quintuple, $< s(t) \in \mathcal{S}, a(t) \in \mathcal{A}, \mathcal{K}, r(s(t), a(t)) \in \mathcal{R}, \gamma >$ where $\mathcal{K}$ is the state transition probability matrix, and $\gamma \in [0, 1]$ is a discount factor for future rewards. $\mathcal{K}$, specifies the probability of being in $s(t+1)$ due to action $a(t)$ taken at state $s(t)$. However, it is difficult to model the state transitions for complex problems such as vehicle dynamics, thus we adopt an actor-critic method, which is model free and exhibits low computational complexity. In the actor-critic framework, the critic uses a function approximator to learn the value function parameters β optimizing the value function ($\mathcal{Q}(s, a|\beta)$), while the actor adopts a function approximator as well to update the policy parameter (η) in the direction suggested by the critic to optimize $\pi(s|\eta)$. In general, RL algorithms employ deep neural networks as function approximators to achieve the optimal solution, and such techniques are collectively called Deep Reinforcement Learning (DRL) methods.

In our work, we use Deep Deterministic Policy Gradient (DDPG) [9], a DRL algorithm that follows the actor-critic framework to learn both the value function and policy. The critic network with parameters β takes care of the value function estimation ($\mathcal{Q}(s, a|\beta)$) while the actor network with parameters η represents the agent's policy ($\pi(s|\eta)$). Notably, the DDPG algorithm supports the continuous action space and suits the 2LL-CACC scheme to learn the optimal ego vehicle acceleration profile. As the name signifies, it learns a deterministic policy where the policy predicts the action directly, rather than predicting a set of probability distributions over the action space $\mathcal{A}$. Since the policy is deterministic, a standard normal noise with zero mean and a standard deviation of 0.1 is added to the predicted action value during training, to ensure the continued exploration of the action space.

Further, it leverages experience replay buffer and target network techniques to ensure a stable and efficient learning process. The experience replay buffer ($\mathcal{Z}$) stores the agent's experience samples as a tuple $(s(t), a(t), s(t+1), r(s(t), a(t)), d(t))$ at every step, where $d(t)$ is a binary value indicating whether the state $s(t+1)$ is a terminal state or not. In the replay buffer $\mathcal{Z}$, the next state ($(s(t+1)$)) is represented as s' as it holds transitions from several time steps. The algorithm then randomly draws the experience samples from the buffer during the learning process. Since the replay buffer allows using the same transitions multiple times, the experience replay buffer improves the sample efficiency and removes the correlation between them by random sampling. The target network, instead, helps stabilize the learning. In general, the value function tries to minimize the Mean-Squared Bellman Error (MSBE), which indicates the difference between the current value function and the value function with greedy policy (taking actions with maximum expected return). It is represented as:

$$L(\beta, \mathcal{Z}) = \mathbb{E}_{\mathcal{Z}}[(\mathcal{Q}(s, a|\beta) - y_t)^2]), \text{with} \tag{9}$$

$$y_t = r(s, a) + \gamma(1 - d) \max_{a'}(\mathcal{Q}(s,' a'|\beta) \tag{10}$$

$$a' = \pi(s|\eta) \tag{11}$$

$$(s, a, s,' r, d) \in \mathcal{Z}\,. \tag{12}$$

Note that both terms in (9) depend on the same value function parameters β. Eventually, it causes instability in the learning process as both the terms in the (9) keep changing. Thus, the structure of the main actor and critic network is cloned as a target actor-and-critic network ($\mathcal{Q}'$ and π') with different parameters (β^*, η^*) to overcome training instability. The target network parameters are used in the (10)–(11) to calculate y_t and later, the MSBE. As the training progresses, the main network parameters (β, η) are gradually updated to the target network (β^*, η^*) through the Polyak averaging technique. Essentially, the critic network is trained to minimize the mean square error of the target network's expected return and value predicted by the critic network, while the actor network aims to maximize the critic network's mean value for the actions predicted by the actor. Subsequently, the model learns to predict the actions with maximum critic value for the current state.

Algoritthm 1: DRL-based Acceleration Control.

```
Randomly initialize critic network $\mathcal{Q}(s, a|\beta)$ and actor
  $\pi(s|\eta)$ with weights $\beta$ and $\eta$
Initialize target network $\mathcal{Q}'$ and $\pi'$ with weights $\beta^* \leftarrow \beta$
  and $\eta^* \leftarrow \eta$
Initialize replay buffer $\mathcal{Z}$
for episode= $1, M$ do
    Receive initial observation state $s(1)$
    for $t = 1, T$ do
    Select action $a(t) = \pi(s(t)|\eta)+$ random normal noise
      according to the current policy and exploration noise
    Execute action $a_t$ and observe new state $s(t+1)$,
      rewards of each component ($r_h(s(t), a(t))$,
      $r_s(s(t), a(t))$, $r_c(s(t), a(t))$), environment status $d(t)$
    Get weights ($x_h, x_s, x_c$) from Context Recognition
      Model
    Calculate the reward $r(s(t), a(t))$ based on the
      weights and their reward components
    Store transition $(s(t), a(t), s(t+1), r(s(t), a(t)),$
      $d(t))$ in $\mathcal{Z}$
    Sample a random mini batch of N transitions
      $(s_i, a_i, s_{i+1}, r_i, d_i)$ from $\mathcal{Z}$
    Set $y_i = r_i + \gamma\mathcal{Q}'(s_{i+1}, \pi'(s_{i+1}|\eta*)|\beta^*)$
    Update critic by minimizing the loss:
      $L = \frac{1}{N}\sum_i(y_i - \mathcal{Q}(s_i, a_i|\beta))^2$
    Update the actor policy using the sampled policy
      gradient:
      $\nabla_\eta J \approx \frac{1}{N}\sum_i \nabla_a \mathcal{Q}(s, a|\beta)|_{s=s_i, a=\pi(s_i)} \cdot$
       $\nabla_\eta \pi(s|\eta)|s_i$
    Update the target networks:
        $\beta^* \leftarrow \rho\beta + (1-\rho)\beta^*$
        $\eta^* \leftarrow \rho\eta + (1-\rho)\eta^*$
  end for
end for
```

2) The DRL-Based Acceleration Control: The DRL-based ACC application we develop seeks to optimally determine the ego vehicle's acceleration through system state information gathered from the ego vehicle's sensors and neighboring vehicles. The pseudo-code of the proposed scheme is presented in Algorithm 1.

States and Action: At a certain time-step t, the state space of the environment is represented by: (i) the lead vehicle acceleration $\alpha(t)$, (ii) the headway $\vartheta(t)$, (iii) the headway derivative $\Delta\vartheta(t)$, (iv) the longitudinal slip $\xi(t)$, (v) the friction coefficient $\mu(t)$, and (vi) the relative velocity $\nu(t)$. In the state space, the preceding vehicle acceleration is obtained through V2X communication, which is simulated with the help of the Co-MoVe framework, and we assume the road friction coefficient is provided by an external estimation method running in the ego vehicle. The headway ($\vartheta(t)$) and longitudinal slip $\xi(t)$ variables are formulated through (3) and (5), respectively. The remaining

TABLE II
CONVERSION VALIDATION

Headway RMSE (Ideal = 1.3 s)		
Validation Models	PyTorch	MATLAB
1	0.1766	0.1799
2	0.184	0.186
3	0.2165	0.2187
4	0.1926	0.1951
5	0.1645	0.1665

state variables are formulated as:

$$\Delta\vartheta(t) = \vartheta(t) - \vartheta(t-1) \tag{13}$$

$$\nu(t) = \mathcal{V}_{lead}(t) - \mathcal{V}_{ego}(t) \tag{14}$$

where $\vartheta(t)$ and $\vartheta(t-1)$ are the headway values at time t and $t-1$ (resp.), and $\mathcal{V}_{ego}(t)$ and $\mathcal{V}_{lead}(t)$ represent ego and lead vehicle's velocity (resp.) at time t. In our model, and in contrast to prior art [13], [15], we also account for the wheel longitudinal slip ratio and road friction coefficient, to represent the vehicle stability.

Since our DRL model aims to control the ego vehicle's acceleration, action $a(t) \in \mathcal{A}$ is defined as a continuous variable. Further, the action values are bounded, in regular conditions, between [-2, 1.47] to provide a comfortable travel experience [33]. The DRL agent receives a numerical value from the environment as feedback on the agent's behavior, a numerical reward that motivates the DRL agent to satisfy the desired objective. The sampling interval of our framework is $\tau = 100$ ms long; the state observation and action decision routine are performed every τ seconds.

Reward Components: The reward function comprises three components: headway (representing traffic flow efficiency), stability (representing safety), and comfort, each component's value ranging in [-1,1]. More formally, we have:

$$r(s(t), a(t)) = x_h \cdot r_h(s(t), a(t)) + x_s \cdot r_s(s(t), a(t)) + x_c \cdot r_c(s(t), a(t)) \tag{15}$$

where x_h, x_s, x_c are the weight coefficients obtained from the Context Recognition Model, which dynamically vary according to the current state, and $r_h(s(t), a(t))$, $r_s(s(t), a(t))$, $r_c(s(t), a(t))$ are, respectively, the headway, vehicle stability, and comfort reward component at time step t. The three reward components are detailed below.

Headway reward component: Headway is a proxy way to measure the gap between two successive vehicles, i.e., ego and lead vehicle [13], [15], and it can be calculating using (3). Following [13], we set the ideal headway to secure a safe and efficient inter-vehicle distance to 1.3 s, while headway values lower than 0.5 s imply a possible risky situation between the ego and the lead vehicle. Traffic efficiency is further ensured by adding the relative velocity between ego and lead vehicle to the headway term, as in (17). The headway term remains unchanged if the ego and lead vehicle travel at the same speed. If instead the ego vehicle travels faster or slower than the lead vehicle, its velocity affects the relative distance, hence the headway. Thus, the addition of the relative velocity helps regulate the ego vehicle's acceleration proactively. Compared to our preliminary work [24], the addition of relative velocity to the state space ($\mathcal{S}$) and reward calculation assists the DRL model to consider the neighboring vehicles traveling at different velocities effectively.

TABLE III
DRL HYPERPARAMETER VALUES

	DDPG	DDQN
Action Space	(-2, 1.47)	[-2.0, -1.6, -1.2, -0.8, -0.4, 0.09, 0.4, 0.8, 1.2, 1.47]
Hidden Layers	3 (actor, critic each)	6
Neurons	64	64
Actor Learning Rate	0.0001	0.0001
Critic Learning Rate	0.001	-
Target Network Update	0.001 (ρ)	100 (steps)
Replay Buffer Size	50000	500000
Mini-Batch Size	48	64

TABLE IV
PARAMETER VALUES

Parameters	Values
$\mathcal{V}_{min}$	$0\,m/s$
$\mathcal{V}_{max}$	$36\,m/s$
τ	100 ms
pi	3.142
M_1	2
M_2	0.4944
M_3	9
M_4	2.0099
M_5	3
M_6	0.5
M_7	0.8
M_8	$6\,ms^{-2}$
M_9	0.35 s

The headway reward component ($r_h(s(t), a(t))$) is modeled as a Log-Normal distribution function with mean ϵ and variance σ, equal to 0.285 and 0.15, respectively:

$$r_h(s(t), a(t)) = M_1 \cdot F_h - 1, \quad \text{with} \tag{16}$$

$$\varphi(t) = \vartheta(t) + \vartheta(t) \cdot \nu_{norm}(t) \tag{17}$$

$$\nu_{norm}(t) = \frac{\nu(t) - \mathcal{V}_{min}}{\mathcal{V}_{max} - \mathcal{V}_{min}} \tag{18}$$

$$F_h = M_2 \cdot f_{lognorm}(\varphi(t)|\epsilon, \sigma), \tag{19}$$

$$f_{lognorm}(x|\epsilon, \sigma) = \frac{1}{\sigma\sqrt{2 \cdot pi}} \exp\frac{(-(\ln x - \epsilon)^2)}{2\sigma^2} \tag{20}$$

where $\mathcal{V}_{min}$, $\mathcal{V}_{max}$, M_1, M_2, pi are the parameters of the headway reward component and their respective values are defined in Table IV. Such headway reward function reaches $+1$ for $\varphi(t) = 1.3$ s, and -1 for $\varphi(t) = 0.5$ s with the specified parameter values.

Comfort reward component: It is associated with the rate of change of acceleration with time, i.e., jerk $j(t)$. According to [33], the best comfort is observed when the absolute jerk value is below 0.9 m/s^3, while values above 1.3 m/s^3 indicate aggressive driving. Therefore, the reward function decreases gradually with the jerk value rising from 0.6 m/s^3 to 2 m/s^3, and it saturates with the minimum reward of -1. To satisfy the desired jerk reward trend, the comfort reward component is modeled using Polynomial Curve Fitting. It is worth noting that the passengers' safety supersedes the comfort factor during critical situations. Thus, we consider the TTC as a safety indicator to identify dangerous situations. The comfort reward is neglected when TTC$\leq$ 4 s, to prioritize safety during such situations. The comfort reward is formulated as:

$$r_c(s(t), a(t)) = \chi(t) \cdot f(jerk), \quad \text{with} \tag{21}$$

$$f(jerk) = \mathit{polyfit}(j(t), M_3) \tag{22}$$

where M_3 is the polynomial degree parameter specified in Table IV, and $\chi(t)$ is the safety indicator declared in (14), which is used to discount comfort in the case of danger.

Stability reward component: It is valued in terms of the slip, i.e., the maximum tractive force of a pneumatic tire on road surfaces. Based on experimental data, an absolute longitudinal slip value below 0.2 is considered a stable condition. Thus, the stability reward gives a maximum reward of $+1$ for zero slip, and a negative reward for slip values over 0.2, indicating that the vehicle is not in the stable region. The stability reward is given by a $\tanh$ function as:

$$r_s(s(t), a(t)) = M_4 \cdot (F_s + 1) \quad \text{with} \tag{23}$$

$$F_s = \tanh(-M_5 \cdot \xi(t)) \tag{24}$$

where M_4, M_5 are scaling parameters and their respective values are reported in Table IV.

Simulation Environment: To learn the desired behavior, the DRL agent has to interact with an environment that simulates the neighboring vehicle's behaviors and road conditions. Our study uses CoMoVe, a comprehensive simulation environment that can accurately simulate all vehicles' dynamics and sensor arrays, V2X communication, and road conditions to facilitate the DRL agent's learning process. Section IV explains in detail the integration of the DRL model with the CoMoVe simulation framework.

Importance of V2X Communication: The role of communication in the 2LL-CACC scheme is crucial as it is responsible for collecting lead vehicle's acceleration to form the state space ($\mathcal{S}$) in the DRL model. Furthermore, in the cut-in and cut-out scenarios, the lead vehicle often falls in the blind spot of the ego vehicle's sensor array, resulting in late detection of the lead vehicle's presence and in uncomfortable maneuvers to avoid a potential collision. Through V2X communications, instead, the ego vehicle can periodically receive information on the lead vehicle's movements (e.g., yaw rate and position) and recognize in advance its intention to change lane, even before the sensor array can perceive it.

To fully benefit from such additional information, we introduce a *gradual switching technique* that allows the ego vehicle to gradually switch the attention to the cut-in vehicle, or the vehicle ahead of the cut-out vehicle, to perform moderate evasive maneuvers without hindering the passengers' safety and comfort. Denoting with $\mathcal{Y}_e$ and $\mathcal{Y}_c$, respectively, the lateral position of the ego vehicle and that of the generic lane-changing vehicle, we define $\Delta\mathcal{Y}_c$ as:

$$\Delta\mathcal{Y}_c = \mathcal{Y}_c - \mathcal{Y}_e \,. \tag{25}$$

Then we let the ego vehicle trigger a lead-vehicle switch whenever $\Delta\mathcal{Y}_c$ crosses a certain threshold. Specifically, in the cut-out scenario, the ego vehicle switches to the new lead vehicle when $\Delta\mathcal{Y}_c > M_6 \cdot \mathcal{Y}_0$ with $\mathcal{Y}_0$ being the lane width (i.e., 3.3 m) and M_6 a scaling factor. In the cut-in scenario, instead, the ego vehicle takes as new lead vehicle the one cutting-in when $\Delta\mathcal{Y}_c < M_7 \cdot \mathcal{Y}_0$ with M_7 being a scaling factor. The values of the scaling factors we used are presented in Table IV. Since the gradual switching indicates slowly shifting the focus from one vehicle to another, this scaled variable suits well our methodology and actual implementation. Next, let us introduce a normalized variable $\wp$ scaled between 0 and 1 according to the specified thresholds.

As long as $\Delta\mathcal{Y}_c$ is less than the threshold value in the cut-out scenario, we compute the headway to be fed to the DRL model as:

$$\vartheta(t) = \frac{\wp\Delta P_c(t) + (1-\wp)\Delta P_n}{\mathcal{V}_{ego}} \tag{26}$$

where n is the new lead vehicle, identified by the ego vehicle based on the values of yaw rate received from its neighbors through V2X communication. Similarly, as long as $\Delta\mathcal{Y}_c$ is greater than the threshold in the cut-in scenario, the headway input to the DRL model is:

$$\vartheta(t) = \frac{\wp\Delta P_p + (1-\wp)\Delta P_c}{\mathcal{V}_{ego}} \tag{27}$$

where p is the previous lead vehicle. Specifically for cut-in situations, the *gradual switching technique* incorporates an adaption of the Automated Lane Keeping System (ALKS), UN Regulation No. 157 [34]. As per the regulation suggestions, the *gradual switching technique* is refined to wait for at least 0.72 seconds before reacting to the cut-in vehicle to avoid considering any temporary lateral position changes in the social vehicle. Subsequently, if the social vehicle's lateral position continues to change for more than the specified threshold, the proposed switching technique will change the focus gradually to the lane-changing vehicle, considering it a cut-in situation. In addition, we monitor the ego vehicle's Time-to-Collision (TTC) concerning the social vehicle and the social vehicle's lateral position during the lane-changing phase to handle aggressive cut-in situations. The ego vehicle will switch its focus entirely to the lane-changing social vehicle if any of the conditions are met:

- TTC becomes lower than the $TTC_{LaneIntrusion}$ [34] threshold, defined as:

  $$TTC_{LaneIntrusion} = \frac{\nu}{2 \cdot M_8} + M_9 \tag{28}$$

  where ν is the relative velocity between the lane-changing social vehicle and the ego vehicle, while M_8 and M_9 are scaling factors accounting for maximum deceleration rate and reaction time, respectively;
- TTC is less than 4 s [32];
- The social vehicle is 30 cm [34] inside the ego vehicle's lane.

The relative velocity between the ego vehicle and the lead vehicle (ν) is computed similarly, and further used by the DRL

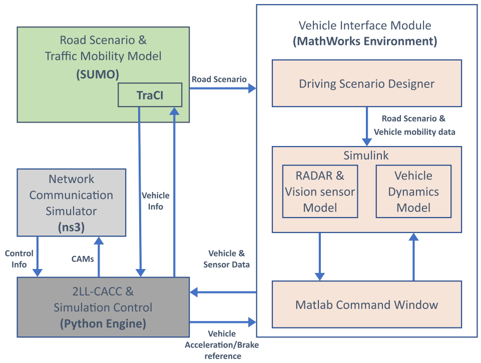

Fig. 3. Architecture of the CoMoVe framework.

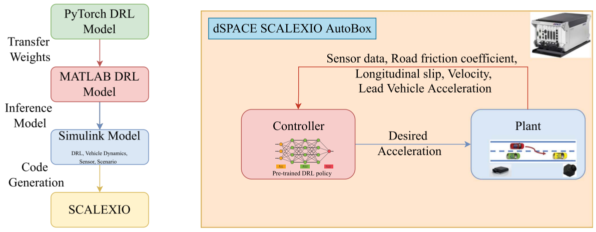

Fig. 4. Process flow of HIL implementation (left); dSPACE SCALEXIO Simulation process (right).

framework to control the ego vehicle movements. In summary, the gradual switching technique is specifically introduced to handle challenging vehicle maneuvers such as cut-in and cut-out. In fact, it influences the DRL model variables to advise the agent to accommodate the lane-changing maneuvers efficiently. The results reported in Section VI-B further validate the importance of V2X communication in the proposed framework.

IV. Integrating the DRL Model in CoMoVe

The CoMoVe framework [30], depicted in Fig. 3, combines widely used simulators in each domain (mobility, communication, and vehicle dynamics) and makes them to interact efficiently. It combines: (i) SUMO, a traffic simulator for vehicle mobility, (ii) ns-3, a network simulator to model V2X communications, (iii) the MATLAB/Simulink module modeling the vehicle dynamics and the vehicle on-board sensors while Driving scenario designer converts the vehicle information from SUMO to MATLAB format to support the on-board sensing, (iv) a Python Engine as a middle-man to handle the information flow between the modules and the host control strategies.

CoMoVe leverages SUMO's TraCI library, ns3's Python bindings, and MATLAB's Python Engine to write complete Python simulation scripts and ensure efficient interactions between them. Consequently, the Python Engine is the CoMoVe's core: it can access information from each simulator and hosts the 2LL-CACC framework to control the ego vehicle movement. As for the DRL state components, the lead vehicle acceleration value ($\alpha(t)$) is received through the ns3 V2X communication model, while the vehicle sensor model output helps calculate the headway ($\vartheta(t)$), headway derivative ($\Delta\vartheta(t)$), and relative velocity $\nu(t)$ values. The longitudinal slip ($\xi(t)$) and friction coefficient ($\mu(t)$) are obtained through the Simulink Vehicle Dynamic model. The DRL model's action (desired acceleration) is used as a reference signal to the ego vehicle's lower level controller in the Vehicle Dynamics Model. A pure electric vehicle with a 14-Degree-of-Freedom (DoF) mathematical model and rear in-wheel motors are utilized to characterize the vehicle dynamics.

Using the CoMoVe framework, in Section VI we show how 2LL-CACC provides a safe, comfortable, and efficient driving experience in challenging road scenarios.

V. Hardware-in-the-Loop Implementation

Testing and validating ADAS subsystems in assembled vehicles incurs significant overhead in terms of time, safety, and cost. Thus, HIL simulations have emerged as a convenient way to virtually validate the system in a wide range of test scenarios during the vehicle development process. In general, HIL simulations validate control algorithms through a real-time virtual environment encompassing the vehicle's functionalities.

To validate our approach, we perform HIL simulations using dSPACE real-time systems comprising modular and robust platforms for testing autonomous driving. Notably, HIL simulations demonstrate the deployable nature of the proposed controller with a similar outcome in the actual vehicle.

More specifically, in the proposed framework, the implementation of HIL simulation involves two main steps:

(i) Conversion of the pre-trained Python DRL model into MATLAB/Simulink supported DRL model, and
(ii) Generation of the DRL agent.

Since the vehicle sensor and dynamics models are simulated in the MathWorks environment, the Python-based DRL agent must be converted into the MATLAB-supported DRL agent for auto code generation. Note that the network simulator (ns3) and traffic mobility model (SUMO) are not part of the HIL implementation, as they do not support the auto code generation process. Instead, the HIL simulation uses the mobility traces of the lead vehicles' and is assumed to be equipped with the vehicle's V2X communication On-Board Unit (OBU) to receive the lead vehicle information. As specified in Section III, we embedded the DDPG algorithm into the CoMoVe framework through the Python Engine. Specifically, the PyTorch machine learning framework is used to build and train the DDPG algorithm's neural network model. In general, the Open Neural Network Exchange (ONNX) format is used to achieve interoperability between different ML frameworks like TensorFlow, PyTorch, and MATLAB. However, the support of the ONNX format in MATLAB is limited to the 3D input layers, i.e., images, so the direct usage of the PyTorch model in MATLAB is unattainable. As a workaround, we replicated the PyTorch neural network structure in MATLAB, and its learnable parameter values are transferred to the MATLAB model. In essence, the learnable parameters are the optimized weights and biases of the neural network that are learned to achieve the desired outcome.

Then, the MATLAB DRL model is converted into a function to evaluate the learned policy of the DRL agent. At a given time step t, the generated function can predict the action $(a(t))$ based on the state $(s(t))$, as per the trained optimal policy. Subsequently, the function is integrated into the Simulink model through the “MATLAB Function” block, so that it can directly predict the control action for the ego vehicle in Simulink. Notice that the generated function does not support further learning and can only be used to perform inference.

Finally, we validated the performance of PyTorch and MATLAB DRL agents by transferring multiple model parameters from PyTorch to MATLAB. Table II presents the observed Root Mean Square Error (RMSE) of the headway parameter concerning the PyTorch and MATLAB DRL agents. The validation results indicate that the effect of the model conversion on the output values is negligible, thus firmly confirming the correct transfer of the PyTorch DRL model to MATLAB. In the second step, the dSPACE’s Real-Time Interface (RTI) links the Simulink software with the dSPACE hardware. In particular, the RTI extends Simulink’s C code generator to execute the Simulink software model in real-time hardware. Later, the generated C code is loaded into the dSPACE SCALEXIO AutoBox to perform the HIL simulation.

dSPACE simulates two main subsystems: Controller and Plant. The controller subsystem provides the desired acceleration for the ego vehicle based on the current states; the Plant subsystem instead simulates the ego vehicle dynamics, sensors, and lead vehicles’ mobility traces. In this work, we used a dSPACE SCALEXIO AutoBox hardware equipped with Intel Core i7-6820EQ, the quad-core processor. The results of the HIL simulations are discussed later in Section VI-B.

Fig. 4 shows the process flow of our HIL implementation (left) and the structure of the HIL simulation platform in dSPACE SCALEXIO (right).

VI. Performance Evaluation

This section first introduces the realistic settings, under which we derive the performance of the 2LL-CACC, as well as the state-of-the-art technique that we consider as benchmark (Section VI-A). Then, using both the CoMoVe framework and the HIL implementation, it presents the obtained performance results in relevant, practical scenarios (Section VI-B).

A. Reference Scenario and Test Cases

We explore two highly challenging highway driving scenarios where a lead vehicle cut in and out from its current lane, exposing the ego vehicle to unclear or critical situations. In both scenarios, as shown in Fig. 5, the lead vehicle (LV), i.e., LV-A in the cut-in and LV-B in the cut-out scenario, is in the sensor array’s blind spot. Thanks to V2X communication, the ego vehicle becomes aware of the LV’s lateral movements and employs the gradual switching technique to change its focus between the two lead vehicles. Note that the ego vehicle’s sensor array takes over the perception control only once the new LV is in its field of view. Fig. 6 illustrates the lateral movement of the LVs in the considered mobility scenario.

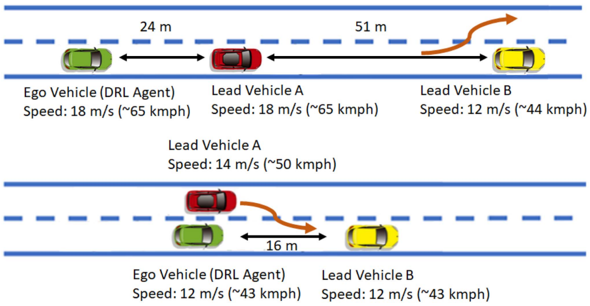

Fig. 5. Cut-out (top) and cut-in (bottom) scenarios.

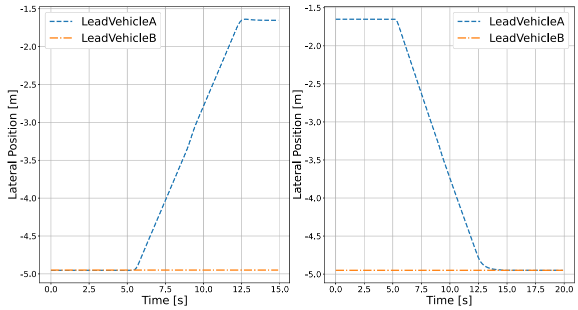

Fig. 6. Lead vehicles’ lateral movement in the cut-out (left) and cut-in (right) scenarios.

In addition, we compare the proposed framework to the state-of-the-art method in [16], whose implementation cannot be entirely reproduced because their dataset is not available for public use. We therefore consider the vanilla DDQN algorithm [35], which is the core component of the method used in [16]. Table III shows the hyperparameter values we used for the DDPG and DDQN algorithms and Table IV presents the parameter values of the different reward components. Section VI-B discusses the behavior of the ego vehicle equipped with the 2LL-CACC, and its relative performance with the case where at the bottom layer we use the state-of-the-art vanilla DDQN algorithm instead of the proposed DRL model, in the challenging cut-in and cut-out maneuvers. For brevity, in the plots shown in the following we refer to the considered benchmark as DDQN.

B. Results

We start by discussing the performance of 2LL-CACC in the cut-out scenario. The top left and top right plots of Fig. 7 present the velocity and acceleration profile of the vehicles. The bottom left and right plots show instead the headway and the jerk trend, i.e., the efficiency and comfort factor of the objectives. In Fig. 7, the black line indicates the ego vehicle’s desired operating range to maintain safe inter-vehicle distance, provide adequate comfort, and improve road usage efficiency. As mentioned, in the cut-out scenario LV-A changes lane at the last moment, to avoid collision with the slow-moving vehicle in front of it. As the ego vehicle monitors the LVs’ lateral movements, it responds to the lane-changing behavior by gradually switching its focus to LV-B.

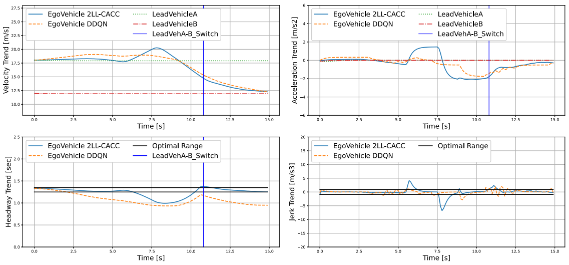

Fig. 7. Cut-out scenario. Left: Vehicles' velocity (top) and ego vehicle's headway (bottom). Right: Vehicles' acceleration (top) and ego vehicle's jerk (bottom), under 2LL-CACC and DDQN.

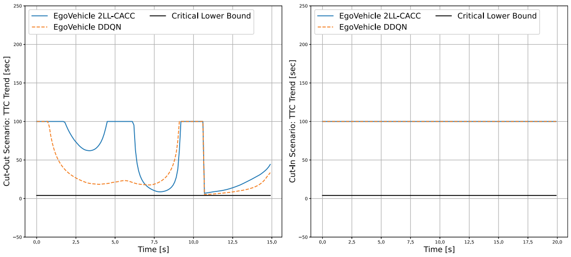

Fig. 8. Time-To-Collision trend of cut-out (left) and cut-in (right) scenarios, under 2LL-CACC and DDQN.

Notice that the ego vehicle's DRL model is designed to maintain a headway of 1.3 s, but the headway increases as the ego vehicle gradually switches its focus to LV-B. Initially, the ego vehicle speeds up to compensate for the rise in headway; however, it also keeps track of the TTC with LV-A to ensure it does not collide with it before it completes the lane-changing maneuver. Once LV-A's lateral position is far enough, LV-B becomes a primary focus for the ego vehicle. Subsequently, the ego vehicle decelerates to maintain zero relative velocity with LV-B, as the latter travels at a lower velocity.

From the bottom left plot of Fig. 7, we can see that 2LL-CACC maintains the headway inside the desired range for about 69% of the simulation time. Also, although the ego vehicle cannot keep the headway inside the desired range during the cut-out maneuver, the left plot of Fig. 8 shows that the TTC never drops below the critical threshold of 4 s, thus always guaranteeing safety. For better visualization, Fig. 8 presents the TTC with an upper bound of 100 s and highlights the lower critical threshold of 4 s with a black horizontal line.

In terms of comfort, one can notice a few spikes in the jerk trend from the bottom right plot of Fig. 7, which however are necessary to maintain in a safe range the TTC between the vehicles, which have clearly higher priority. Also, the context recognition model consider the safety indicator (χ) as one of the input features to appropriately assign weights to the reward components, so as to prioritize passenger safety. The blue line in the figure denotes the time when the sensor array detects the presence of the new LV. One can infer that the detection is indeed very late, and it could lead to unsafe conditions if the ego vehicle responded to the new LV just from that moment.

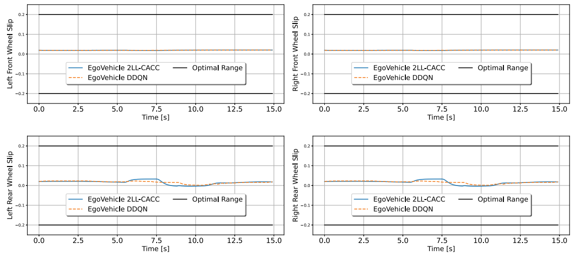

Fig. 9. Cut-out scenario: Vehicle wheel slip under 2LL-CACC and DDQN. Left: left front wheel (top) and left rear wheel (bottom). Right: right front wheel (top) and right rear wheel (bottom).

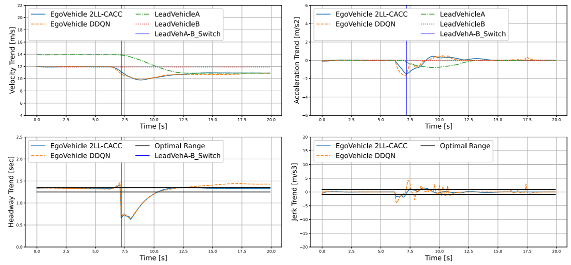

Fig. 10. Cut-in scenario. Left: Vehicles' velocity (top) and ego vehicle headway (bottom). Right: Vehicles' acceleration (top) and ego vehicle jerk (bottom), under 2LL-CACC and DDQN.

As we can see from Fig. 7, the DDQN-assisted ego vehicle does not provide satisfactory results compared to the 2LL-CACC. The bottom left plot of Fig. 7 highlights that the DDQN model maintains the headway inside the desired range only for 15% of the total simulation time, while the counterpart maintains it for 69% of the time, providing much better road usage efficiency. Regarding comfort, DDQN could keep the jerk within the desired range for most of the time, but one can notice the oscillatory behavior showing a frequent change in the acceleration and, hence, resulting in sub-optimal performance. In particular, the DDQN suffers from the usage of discrete action space as in this case the ego vehicle has only a limited set of accelerations to choose from. Since the vehicle travels on a dry road and according to a non aggressive driving behavior, Fig. 9 confirms that the ego vehicle's longitudinal slip is always within the desired range, thus ensuring the vehicle's stability.

Further, it is worth mentioning that we tested the DDQN model with an extended set of actions comprising 19 equally spaced actions between -2 m/s^2 and 1.47 m/s^2. However, the DDQN model could not converge even after 480 episodes since the larger action space increases the model complexity and demands more time to learn the optimal behavior. In contrast, 2LL-CACC learns an optimal policy within 340 episodes and delivers better results than the DDQN.

We now move to the cut-in scenario. The top plots of Fig. 10 show the velocity (left) and acceleration (right) trends of the vehicles' involved in the scenario. The lead vehicle (LV-A) traveling at higher speed overtakes the ego vehicle and then starts a cut-in maneuver to enter the ego vehicle's lane. Also, LV-A decelerates in order to squeeze into the gap between the ego vehicle

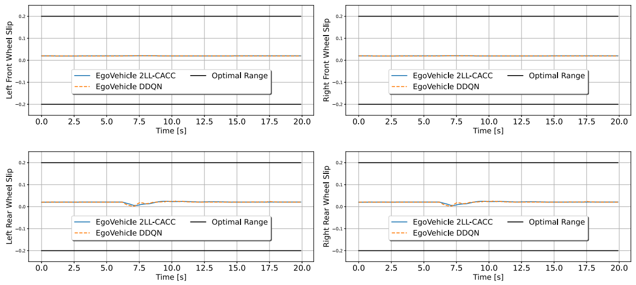

Fig. 11. Cut-in scenario: Vehicle wheel slip under 2LL-CACC and DDQN. Left: left front wheel (top) and left rear wheel (bottom). Right: right front wheel (top) and right rear wheel (bottom).

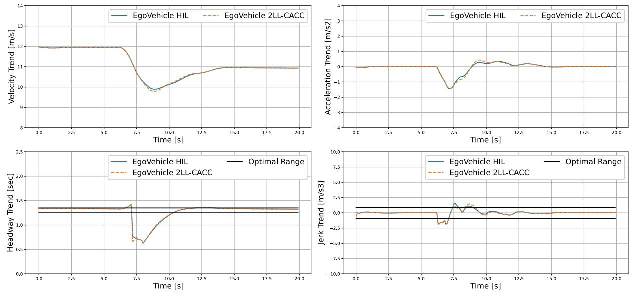

Fig. 12. Cut-in scenario. Left: Ego Vehicle velocity (top) and headway (bottom). Right: Acceleration (top) and jerk (bottom), under 2LL-CACC and HIL.

and LV-B. As before, the ego vehicle promptly recognizes the lane-changing maneuver thanks to V2X communications, and it starts to monitor the lateral movement of the social vehicle. Once the gradual switching determines it is a cut-in situation, the ego vehicle starts decelerating as the distance between them reduces rapidly. Note that in this situation, the distance between the ego vehicle and LV-A does not represent the gap between them. Instead, the relative distance is calculated according to the turning angle and length of LV-A, as it also incorporates the ego vehicle's collision point on LV-A. The calculated relative distance gives additional time to the ego vehicle to handle the maneuver effectively. The camera sensor quickly identifies the LV-A presence and takes control to define the relative distance between the cars. Nevertheless, the role of V2X communication is still crucial, as it recognizes the maneuver proactively and allows the ego vehicle to decelerate gradually.

As for the headway, the bottom left plot of Fig. 10 shows that the 2LL-CACC can maintain such metric within the desired range for a more extended time period compared to the DDQN (78% versus 45% of the total simulation time). Also, the right plot of Fig. 8 shows that the TTC never drops below the critical safety threshold of 4 s: this shows that, even in the closer cut-in situation, the gradual switching can assist the ego vehicle to ensure safety and better road usage efficiency.

The bottom right plot of Fig. 10 underlines that the jerk is temporarily outside the desired range during the cut-in maneuver, but this is again inevitable, as the scenario demands such a response to maintain a safe distance between the cars for the whole simulation period. In this scenario the DDQN model cannot handle the cut-in maneuver as efficiently as the 2LL-CACC. Furthermore, the ego vehicle's longitudinal slip presented in Fig. 11 shows that the vehicle remains stable with both DDPG- and DDQN-based models, given that the cut-in scenario is carried out on dry road conditions.

In addition, we have executed the cut-in scenario in the dSPACE Scalexio AutoBox and verified the HIL system's performance. As can be seen in Fig. 12, the HIL implementation achieves similar performance to that obtained with the standard model in the loop setup. This result further strengthens the 2LL-CACC's ability, as it validates the deployable nature of the trained DRL model in actual vehicles.

Finally, Table V presents the Root Mean Square Error (RMSE) of the obtained results to highlight the quantitative performance of the proposed framework. 2LL-CACC achieves very good results with respect to all objectives, and notably outperforms the DDQN-based model in all scenarios. In terms of comfort, 2LL-CACC has higher jerk RMSE values; however this is inevitable, to promptly react to new conditions, as passengers' safety has to be prioritized over comfort in these critical scenarios. Still, 2LL-CACC achieves an excellent trade-off among the three objectives, providing safe inter-vehicle distance, improved road usage efficiency, and satisfactory comfort to the passengers.

TABLE V
Comparison Between 2LL-CACC and DDQN in Terms of RMSE for Headway, Jerk, and Slip

RMSE			
Metrics	Scenarios	2LL-CACC	DDQN
Headway	Cut-out	0.1184	0.2515
(Ideal = 1.3 s)	Cut-in	0.1767	0.1872
Jerk	Cut-out	1.2405	0.6976
(Ideal = $0\, m/s^3$)	Cut-in	0.4715	0.8812
Slip	Cut-out	0.018	0.018
(Ideal = 0)	Cut-in	0.02	0.02

VII. Conclusion

We addressed Adaptive Cruise Control (ACC) for connected autonomous vehicles in such challenging traffic scenarios as the cut-in and cut-out maneuvers. We proposed a 2-layer, ML-assisted deep reinforcement learning (DRL) approach that weighs the target metrics such as headway, jerk, and longitudinal wheel slip properly and achieves the best trade-off among safety, road efficiency, and comfort objectives. When compared with state-of-the-art alternatives, our framework provides substantially better performance. Notably, it achieves 54% and 33% better headway than its alternatives, thus ensuring better traffic flow efficiency. Particularly, the V2X communication enables the ego vehicle to be timely and gradually switch its focus to the neighboring vehicles, significantly boosting safety performance. While we have considered roads to be straight (hence, lane-changing maneuver only influences the lead vehicle's yaw rate), future work will leverage ADAS applications like lane change detectors and extend the proposed framework to turning roads.

Acknowledgment

The Views Expressed are Those of the Authors and Do Not Necessarily Represent the Projects.

REFERENCES

[1] Health Effects Institute, Traffic-related air pollution: A critical review of the literature on emissions, exposure, and health effects: Executive summary. [Online]. Available: https://www.healtheffects.org/system/files/SR17TrafficReview_Exec_Summary.pdf

[2] INRIX, "Inrix global traffic scorecard - U.K," [Online]. Available: https://inrix.com/press-releases/2019-traffic-scorecard-UK

[3] S. Kim et al., "Analysis of human driver behavior in highway cut-in scenarios," in *WCX: SAE World Congr. Experience*, Mar. 2017, pp. 1–2.

[4] E. Thorn, S. C. Kimmel, and M. Chaka, "A framework for automated driving system testable cases and scenarios," Sep. 2018, dOT HS 812 623. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/38824

[5] A. Maurya and S. Das, "Time headway analysis for four-lane and two-lane roads," *Transp. Developing Economies*, vol. 3, pp. 1–18, Apr. 2017.

[6] Y. Li, H. Lu, X. Yu, and Y. G. Sui, "Traffic flow headway distribution and capacity analysis using urban arterial road data," in *Proc. IEEE Int. Conf. Electric Technol. Civil Eng.*, 2011, pp. 1821–1824.

[7] A. Shrivastava and P. Li, "Traffic flow stability induced by constant time headway policy for adaptive cruise control (ACC) vehicles," in *Proc. IEEE Amer. Control Conf.*, 2000, vol. 3, pp. 1503–1508.

[8] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[9] T. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[10] EuroNCAP, "2018 automated driving tests." [Online]. Available: https://www.euroncap.com/en/vehicle-safety/safety-campaigns/2018-automated-driving-tests/

[11] B. Sultan, M. Brackstone, B. Waterson, and E. R. Boer, "Modeling the dynamic cut-in situation," *Transp. Res. Rec.*, vol. 1803, no. 1, pp. 45–51, Jan. 2002.

[12] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.

[13] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1248–1260, Dec. 2011.

[14] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2019, pp. 2326–2331.

[15] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transp. Res. Part C: Emerg. Technol.*, vol. 117, 2020, Art. no. 102662.

[16] Q. Chen, W. Zhao, L. Li, C. Wang, and F. Chen, "ES-DQN: A learning method for vehicle intelligent speed control strategy under uncertain cut-in scenario," *IEEE Trans. Veh. Technol.*, vol. 71, no. 3, pp. 2472–2484, Mar. 2022.

[17] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 1379–1384.

[18] A. Bonfitto, S. Feraco, A. Tonoli, and N. Amati, "Combined regression and classification artificial neural networks for sideslip angle estimation and road condition identification," *Veh. Syst. Dyn.*, vol. 58, no. 11, pp. 1766–1787, 2020. [Online]. Available: https://doi.org/10.1080/00423114.2019.1645860

[19] Y. Zhao, H. Li, F. Lin, J. Wang, and X. Ji, "Estimation of road friction coefficient in different road conditions based on vehicle braking dynamics," *Chin. J. Mech. Eng.*, vol. 30, pp. 982–990, 2017.

[20] S. Rajendran, S. K. Spurgeon, G. Tsampardoukas, and R. Hampson, "Estimation of road frictional force and wheel slip for effective antilock braking system (abs) control," *Int. J. Robust Nonlinear Control*, vol. 29, no. 3, pp. 736–765, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.4366

[21] A. Carvalho, A. Williams, S. Lefevre, and F. Borrelli, "Autonomous cruise control with cut-in target vehicle detection," in *Proc. Int. Symp. Adv. Veh. Control*, 2016, pp. 93–98.

[22] V. Milanes and S. Shladover, "Handling cut-in vehicles in strings of cooperative ACC vehicles," *J. Intell. Transp. Syst.*, vol. 20, pp. 1–14, Feb. 2015.

[23] C. Chen, J. Guo, C. Guo, C. Chen, Y. Zhang, and J. Wang, "Adaptive cruise control for cut-in scenarios based on model predictive control algorithm," *Appl. Sci.*, vol. 11, no. 11, 2021, Art. no. 5293.

[24] D. C. Selvaraj, S. Hegde, N. Amati, C. F. Chiasserini, and F. Deflorio, "A reinforcement learning approach for efficient, safe and comfortable driving," *24th EURO Work. Group Transp. Meeting*, 2021.

[25] Y. Lin, J. McPhee, and N. L. Azad, "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 221–231, Jun. 2021.

[26] G. Matheron, N. Perrin, and O. Sigaud, "Understanding failures of deterministic actor-critic with continuous action spaces and sparse rewards," in *Proc. Artif. Neural Netw. Mach. Learn.*, 2020, pp. 308–320.

[27] F. Memarian, W. Goo, R. Lioutikov, S. Niekum, and U. Topcu, "Self-supervised online reward shaping in sparse-reward environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2369–2375.

[28] Z. Hu and D. Zhao, "Adaptive cruise control based on reinforcement leaning with shaping rewards," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 15, no. 3, pp. 351–356, 2011.

[29] Y. Ye, X. Zhang, and J. Sun, "Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment," *Transp. Res. Part C: Emerg. Technol.*, vol. 107, pp. 155–170, 2019.

[30] D. C. Selvaraj, S. Hegde, C. F. Chiasserini, N. Amati, F. Deflorio, and G. Zennaro, "A full-fledge simulation framework for the assessment of connected cars," in *Proc. Transp. Res. Procedia*, 2021, vol. 52, pp. 315–322.

[31] G. Louppe, "Understanding random forests: From theory to practice," 2014, *arXiv:1407.7502*.

[32] Economic Commission for Europe, "Proposal for a new draft un regulation on the approval of motor vehicles with regard to their advanced emergency braking system for M1 and N1 vehicles," 2019. [Online]. Available: https://unece.org/DAM/trans/doc/2019/wp29/ECE-TRANS-WP29-2019-61e.pdf

[33] I. Bae et al., "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," 2020, *arXiv:2001.03908*.

[34] Economic Commission for Europe, "Proposal for a new un regulation on uniform provisions concerning the approval of vehicles with regards to automated lane keeping system," 2020. [Online]. Available: https://documents-dds-ny.un.org/doc/UNDOC/GEN/G20/087/82/PDF/G2008782.pdf

[35] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

Dinesh Cyril Selvaraj is currently working toward the Ph.D. degree in communication networks with the Politecnico di Torino, Turin, Italy.

Shailesh Hegde is currently working toward the Ph.D. degree in mechanical engineering with the Politecnico di Torino, Turin, Italy.

Nicola Amati is currently a Professor of mechanical engineering with the Politecnico di Torino, Turin, Italy.

Francesco Deflorio is currently Professor of transport engineering with the Politecnico di Torino, Turin, Italy.

Carla Fabiana Chiasserini (Fellow, IEEE) is currently a Professor with the Politecnico di Torino, Turin, Italy, an EiC of *Computer Communications*, an Ediot-at-Large of the IEEE/ACM TRANSACTIONS ON NETWORKING and a Member of the Steering Committee of ACM MobiHoc and the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.