International Journal of Serious Games

ISSN: 2384-8766
https://journal.seriousgamessociety.org/

Article

Implementing Deep Reinforcement Learning (DRL)-based Driving Styles for Non-Player Vehicles

Luca Forneris[1], Alessandro Pighetti[1], Luca Lazzaroni[1], Francesco Bellotti[1], Alessio Capello[1], Marianna Cossu[1], and Riccardo Berta[1]

[1]*Department of Electrical, Electronic and Telecommunication Engineering (DITEN), University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy*
{luca.forneris, alessandro.pighetti, luca.lazzaroni, alessio.capello, marianna.cossu}@edu.unige.it ; {francesco.bellotti, riccardo.berta}@unige.it}

Keywords:
Reinforcement Learning
Serious Games
Autonomous Agents
Racing Games
Driving Styles
Decision Making

Received: May 2023
Accepted: November 2023
Published: November 2023
DOI: 10.17083/ijsg.v10i4.638

Abstract

A realistic representation of the traffic participants is a key feature of serious games for driving. We propose a novel method for improving the behavioral planning of non-player vehicles (NPVs) by supporting implementation of (i) "human-like", high-level decision making and (ii) different driving styles. The method relies on the deep reinforcement learning (DRL) technology, which is gaining ever more interest in the real-world automated driving research. We designed a system architecture including advanced driving assistance systems (ADAS) and trained the agent in the highway-env DRL environment. Compared to a low-level decision making system, our system performs better both in terms of safety and speed. Moreover, the proposed approach allows reducing the number of training steps by more than one order of magnitude. This makes the development of new models much more efficient. As a second main contribution, we demonstrate that it is possible to train agents characterized by different driving behaviors, by tweaking the weights of the factors of a general DRL reward function. This approach avoids heuristic coding of the driving styles, saving development and maintenance time. The developed agent models can then be seamlessly deployed as NPVs in any target SG in the same development environment (i.e., highway-env). Furthermore, the information and lessons learned presented in this article can be useful to effectively and efficiently train similar agents in their target SG environment.

1. Introduction

A significant design issue for serious games (SGs) for car driving (e.g., [1]) is given by learning transfer, since playing a game is very different from driving a 1+ ton vehicle with several lives continuously at stake. However, important aspects can be learned or at least experienced also through simulation/gaming, and it is important that SGs are well-designed to meet their specific instructional target. One key aspect in this context is given by the modeling of non-player

vehicles (NPVs), which constitute the traffic around the vehicle driven by the player, namely the ego vehicle (EV). NPVs play a crucial role in SGs for traffic instruction [2] and safety awareness [3], as traffic is a key context of the simulation [4], [5].

NPVs are completely automated vehicles, that continuously take decisions during the game to simulate the traffic. Decision-making is necessary for the behavioral planning of a vehicle, which includes path/trajectory planning and tracking (e.g., [6], [7]). An emerging technology for implementing decision making tasks is deep reinforcement learning (DRL). DRL consists in using deep neural networks (DNNs) as the models mapping observations to actions that are trained through reinforcement learning (RL). RL is a branch of machine learning (ML) in which an agent (i.e., the model) learns, through experience, to take actions in an environment in order to maximize the cumulative reward in an episode [8], [9]. Based on literature about real-world automated driving (e.g., [10]–[14]), we argue that DRL could be useful to develop realistic NPVs for SGs, also faithfully implementing different driving styles (e.g., standard, comfort, aggressive), for a more realistic and difficulty-level-adaptable representation of the traffic surrounding the ego vehicle.

An RL agent learns by interacting with the environment in which it is immersed. Interactions consist of observations and actions. An action is a decision taken by the agent based on the latest observation provided by the environment. Agents are typically trained in configurable simulation environments. Highway-env is a collection of simple 2D bird-eye view environments for training RL agents in automated driving and tactical decision-making tasks, particularly oriented to highway settings [15], [16], even if other car-related scenarios are available as well (e.g., intersection, roundabout, parking lots). Highway-env, developed by Edouard Leurent [17], is one of the environments provided within Farama Foundation Gymnasium (previously Gym [18]), an open-source Python library for developing and comparing RL algorithms by providing a standard application programming interface (API) to communicate between learning algorithms and driving environments. Highway-env directly supports high-level concepts such as vehicles, lanes, etc., and offers several pre-defined simulation environments representing different driving scenarios, such as driving on a highway, navigating a roundabout, parking, etc... In addition to the learning agent (i.e., the EV), a highway-env environment can also be easily populated by concurrent NPVs and customized according to special constraints and requirements, e.g., by defining the number of NPVs within a simulation, specifying the spacing between them or their density.

Highway-env provides two alternative types of low-level input interface to an RL model: one based on discrete actions (faster, slower, left, right, idle) and one based on continuous actuators (throttle level, with negative values indicating the braking; and steering angle). However, state-of-the-art vehicular systems involve higher-level advanced driving assistance systems (ADAS), such as the Adaptive Cruise Control (ACC). Thus, in this work, we are interested in exploring the behavior of NPVs driven by a higher-level RL-based decision maker, which exploits the above mentioned ADASs. Through the development choice of targeting higher-level decision making, we expect to (i) achieve a more realistic behavior of state-of-the-art vehicles, (ii) better reflect the human reasoning when driving a car, and (iii) get benefits in terms of model training, which is the most critical aspect of any machine learning (ML) algorithm, particularly in DRL. In fact, the action space of higher-level decisions may be much narrower than that of the above-mentioned discrete actions or continuous actuators, potentially implying a significant reduction in training time, which would be very useful if a game designer would have to train different agents for different driving styles. Particularly, as we will see in the next section, we are interested in three actions: “keep lane” (i.e., ACC), “overtake” and “go to the right-most lane” (typically after an overtake).

The goal of this article is to study the development through DRL of high-level DM agents (i.e., able to exploit state of the art ADAS) implementing different driving styles, that can be

then employed in SGs as NPVs for a realistic and instructionally useful implementation of the traffic.

The remainder of the article is organized as follows: Section 2 describes the current state of the art related to the topic, Section 3 presents the environment used and introduces the reinforcement learning paradigm, Section 4 is devoted to the presentation of the experiment performed, Section 5 summarizes the experimental results obtained, and finally Section 6 concludes.

2. Related Work

While a large number of games have been developed in the car racing area, SGs to support specific aspects of automotive driving are much rarer. The closeness between the virtual world of the SG and the real world should guarantee realism and user immersiveness to be effective.

The Good Drive game [1] presents players with the major sequences that cover the entire French driving license training program. The scenes involve passing and overtaking in rural and urban areas, and at different times of the day and weather conditions. The player must control his vehicle, follow the road rules, and adapt to what is happening in the environment. The presentation website stresses that Good Drive "does not aim to replace traditional methods for learning to drive; instead, it serves to support these". Car Driving School Simulator [19] is a popular 3D commercial game for driving training before going on the road, with the main goal of learning to respect traffic laws, use turn signals, etc.

Gounaridou et al. [3] present the implementation of an educational game on traffic behavior awareness through the main stages of analysis, design, development, and evaluation. Reported results reveal that a properly developed educational game could enhance traffic awareness through experiential and mediated learning. Likitweerawong et al. [20] present a virtual reality SG providing players with the knowledge of the basic rules before they drive a real car on a real road. User tests revealed that the game can combine enjoyment with learning basic driving elements. Hrimech et al. [21] investigate the effects of the use of SG in eco-driving training. Results demonstrate that the serious game influences positively the behavior of inexperienced drivers in ecological driving.

Variability is not an insignificant factor when it comes to games. It refers to the ability to provide the user with multiple gaming experiences, to avoid the game being constant and repetitive. In the SG context variability plays a major role in making games as realistic and believable as possible [22]. Kichmeier et al. [23] investigate content personalization techniques to add variability into SGs, in order to adapt the interactive application to the player and then provide the possibility of explorative learning processes. Zielke et al. [24] state that believability through natural actions and intelligent interaction is the primary purpose of non-player characters (NPCs), analogously to NPVs in driving and racing games. Therefore, the addition of different driving styles in driving and racing SGs becomes an important tool for adding variability and realism within the game. Massoud et al. [25] propose an algorithm to profile the driving style of the player to provide feedback in order to keep the driver aware of the safety and the fuel economy, highlighting that an aggressive driving style may be more fuel-consuming prone as well as crash-prone. Sagberg et al. [26] outline a tentative framework for understanding the definition of driving style, proposing it as a composition of individual driving skills and personality characteristics, social context, and cultural values, along with the technology of the vehicle itself. Results demonstrate the multidimensionality and complexity of the concept of driving styles. Bellotti et al. [27] illustrate a serious game whose goal is to reward and coach the driver after processing vehicular data. The relative score is displayed on the automotive dashboard, and it is designed to enhance and improve driving skills and behavior. Troullinos et al. [28] enrich the SUMO simulator [29] by first extracting patterns and features from real participants and then creating a virtual population of drivers, each with its

driving style. The pool of drivers may be generated according to population behavior distribution that is more appropriate to a given problem or scenario. This procedure may be used to improve the overall realism of traffic simulations.

Massoud et al. [30] present a set of fuzzy logic models that process signals from basic vehicular sensors (e.g., speed and throttle position) in order to estimate fuel consumption, which is then usable in different ways in reality-enhanced SGs. As a complementary tool, Edgine supports a smart configuration of limited-resource edge devices in order to send the proper information to a cloud-based measurement management system [31]. Westera et al. [32] provide an important general resource, as it provides a comprehensive overview of artificial intelligence (AI) for serious games. Particularly, it presents a set of advanced game AI components that enable pedagogical affordances and that can be easily reused across game engines and platforms. All components have been applied and validated in SGs tested with real end-users. In the specific area of driving games, Tomilson and Melder [33] describe at a high level the requirements, architecture, and best practices for high-speed vehicle racing AI. Perot et al. [34] exploit the WRC6 rally game, with realistic physics and graphics, to train an Asynchronous Actor-Critic (A3C) reinforcement learning (RL) model in an end-to-end fashion. The authors also propose an improved reward function to learn faster. Fakhry [35] presents a Deep Q-Network (DQN) developed for driving a car in a simple racing game.

As the literature lacks information about DRL training of NPVs exhibiting different driving styles, our goal is to advance the state of art by investigating this field also by exploiting "human-like", high-level DM strategy to increase realism and, as a side effect, ease of development.

3. Reinforcement Learning and highway-env development environment

RL is one of the three main paradigms of ML, besides Supervised and Unsupervised Learning. The goal of RL, as shown in Figure 1, is to train an agent, which consists in making it learn a policy to maximize the outcome of its actions applied to an uncertain dynamic system. Since the true relationship between actions and the state of the dynamic system is unknown, the agent is trained with an empirically approximated outcome function called the Reward Function (RF). The training consists of a Monte Carlo trial-and-error approach, usually in a simulated environment, where the agent can explore different strategies and get the corresponding outcome given by the RF. Thus, the RF has a key role in shaping the agent's behavior. When a model is learned using a Deep Neural Network, this paradigm is called Deep Reinforcement Learning (DRL)[36].

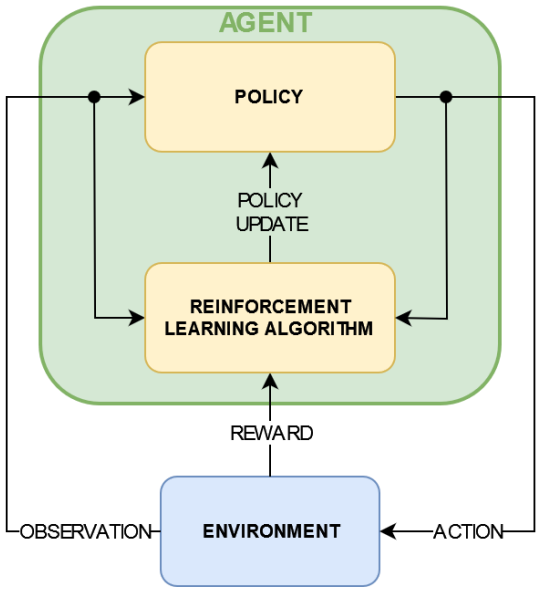

Figure 1. The RL schema.

A variety of algorithms have been developed for training DRL models. They can be roughly divided into three classes:

- Value-based methods (e.g., Q learning [37] and DQN), that choose the next action of an agent by estimating the cumulative benefit of each possible action.
- Policy-based methods, including policy gradient methods, that optimize a performance objective (typically the expected cumulative reward) through gradient ascent [38].
- Model-based methods, which provide a model of the environment to predict how it responds to the agent's actions [39].

For the online training procedure of our models, we used the Proximal Policy Optimization (PPO) algorithm [40]. The PPO is a policy gradient method and supports discrete action spaces, as in our case ("keep lane", "overtake", "right-most lane"). Motivated by the shortcomings of other popular policy gradients algorithms, such as Trust Region Policy Optimization (TRPO) [41] and A2C [42], which suffered from training stability issues and slow policy convergence, PPO introduces smaller policy update steps and a clipped objective function to ensure stable, generalized and efficient learning. Thanks to these features and its implementation simplicity, PPO has quickly gained popularity in a wide range of DRL-related research fields, including game-playing and robotic control [43]–[45].

Our development relies on the Highway-env environment, which exploits Stable-Baselines3 [46] as a DRL support library. Regarding the DRL algorithm, we use the state-of-the-art Stable-Baselines3 PPO implementation [40]. The environment represents a multi-lane highway (Figure 2), where the ego vehicle (EV) is controlled by a DRL agent and has the goal of covering as much distance as possible, avoiding collisions with NPVs.

To avoid a likely source of confusion, it is important to highlight that here we are speaking of NPV in the training environment, while the EV is the subject of the training. On the other hand, once the EV has been trained, it will be usable as a realistic NPV in an SG, as explained in the Introduction.

NPVs in the training environment follow a heuristic behavior, with some randomness applied in order to avoid over-fitting on the agent's part [47]. We also added the number of vehicle randomization at each episode and modified the behavior of the NPVs to improve realism and the EV road navigation. In particular, NPVs drive at a different speed depending on their current lane (slower on the right lane, faster on the left). The discrete lane change

decisions are given by the *Minimizing Overall Braking Induced by Lane Changes* (MOBIL) model [17], [48].

The RF is key to shaping the behavior of an agent, for instance by penalizing high or low speed, or frequent changes of the lane. Thus, a main goal of our research is to explore the shaping and tweaking of different RFs to represent different driving styles, as described in the next section.

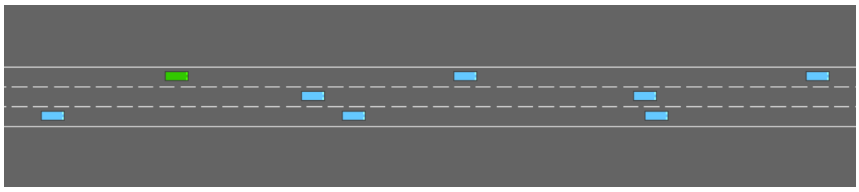

Figure 2. Snapshot from highway-env. The EV is colored green; NPVs are light blue.

4. Experiment

This chapter illustrates our experiment to train agents featuring different driving styles. We first outline the dynamic task environment, then the state observed by the agent and its action's space, and finally, the RF and its parameter values that we propose to achieve the desired differentiation in driving behavior.

4.1 Environment

The adopted training environment, depicted in Figure 2, is composed of an EV and several NPVs, whose number is chosen randomly at the beginning of each episode between 15 and 20, that travel along a 3-lane straight highway. The above number of NPVs has been chosen to simulate an intense but flowing highway traffic, with an average of about 7 to 8 vehicles ahead of the EV, in a 0-100 m. range, at any time. The maximum allowed speed is 36 m/s (130 km/h). According to the default environment settings, a vehicle cannot drive outside the roadway. The RF, as well as many different environment parameters, can be modified through the default Gym configuration interface by overriding the 'env.config.update' method and specifying the desired key-value pairs[49].

4.2 Observations

The observations provided by the environment are the 5 default features defined in highway-env: presence (whether a vehicle is present or not in the current scene), longitudinal and lateral position and velocities; for the EV and the V closest vehicles, where V is a parameter, set to 6 in our case. The employed observation (i.e., kinematic observation [17]) is a V * F matrix, where F is the number of observed features. Hence, the considered kinematic observation matrix is a 7 * 5 matrix in our case, of which an example is provided in Eq. (1):

$$K = \begin{bmatrix} 1 & 1.00 & 0.00 & 0.45 & 0.00 \\ 1 & 0.19 & 0.35 & -0.22 & -0.01 \\ 1 & 0.41 & 0.33 & -0.11 & 0.00 \\ 1 & 0.46 & 0.67 & -0.22 & 0.00 \\ 1 & 0.93 & 0.67 & -0.20 & 0.00 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

The first row always refers to the EV, while the remaining rows describe, in ascending order of Euclidean distance, the V closest vehicles. The observation is normalized within a fixed

range: x and y within [-100, 100] and v_x and v_y within [-20, 20]. The coordinates are relative to the EV, except for the EV itself, which stays absolute. If less than V vehicles are observed in the current scene, the last rows are filled with zeros.

4.3 Action space

Our approach involves raising the level of the agent decision making (DM), by keeping into account availability of state-of-the-art ADAS in vehicles (e.g., the ACC). Thus, we built a decision hierarchy where the EV takes high-level decisions while the ADAS (of which we developed a simple model implementation) translates such decisions into the low-level agent action space (i.e., "faster", "slower", "lane right", "lane left", "idle"), which is implemented in the highway-env. Particularly, we define the following discrete high-level actions for the decisions of the agent:

- Keep lane: the ACC is activated. If a vehicle is present ahead in the lane, the EV keeps a safe time headway. Otherwise, it goes at the maximum speed.
- Overtake: change lane to the left so as to make an overtake if a vehicle is present in the ego lane ahead. Otherwise, the Cruise Control is activated.
- Go to the rightmost lane: change the lane to the right, wait one second, and repeat this, until the rightmost lane is reached.

Figure 3 provides a complete overview of the system. Figure 4 shows the vehicular behaviors enacted by the proposed actions. In our design, the EV includes serial subsystems organized in hierarchy [50]: the higher-level Decision Maker (DM) and the lower-level Behavior Executor (BE). The DM gets the observations from the environment and selects the appropriate high-level action accordingly.

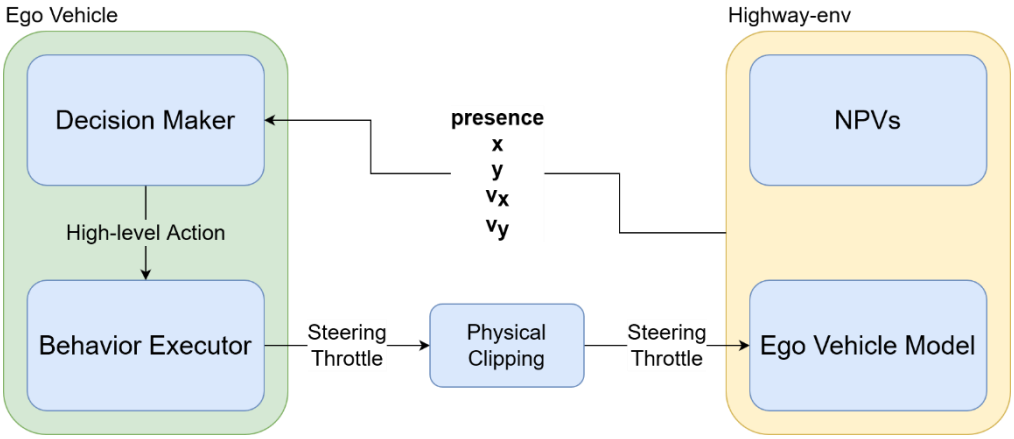

Figure 3. High-level schema of the overall system architecture.

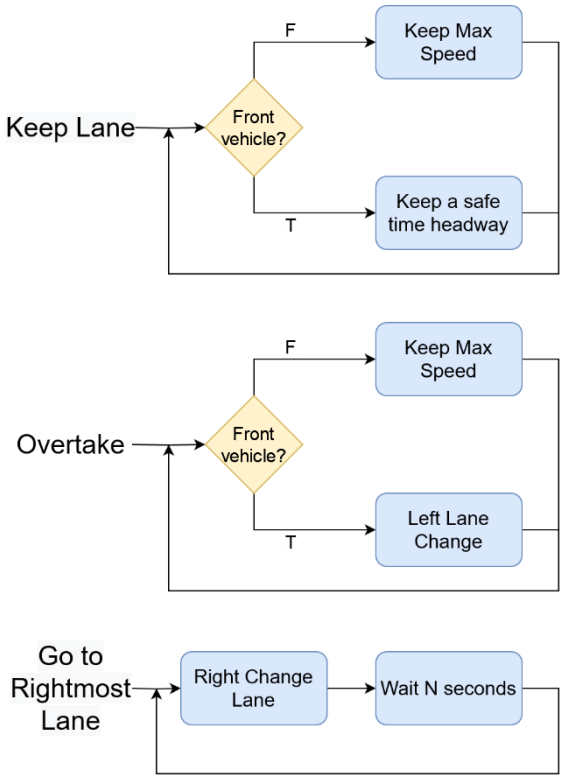

Figure 4. Flowcharts of the high-level behaviors.

The high-level actions are received by the BE, which implements the appropriate behavior as a series of low-level controls (throttle levels and steering angles). Such values are clipped to enforce physical feasibility before feeding the environment, whose interface is left unchanged. Here we stress that this architecture can implement different behaviors, corresponding to different driving styles. For instance, the DM might indicate the EV to left-overtake vehicles at the maximum possible speed in case of aggressive driving behavior, while a more conservative EV might prefer following the vehicle in front, switching on the ACC.

Each behavior is executed in a loop, which is terminated as soon as the DM chooses a different action than the previous one. The BE implementation simulates the behavior of state-of-the-art onboard ADAS through simple proportional integrative derivative (PID) controllers in a closed-loop control schema.

For deriving the acceleration and steering command, we used a straightforward bicycle kinematic model of the vehicle [51], detailed in Eq. (2):

$$\begin{aligned} \dot{x} &= \mathrm{v}\cos(\psi + \beta) \\ \dot{y} &= \mathrm{v}\sin(\psi + \beta) \\ \dot{v} &= a \\ \dot{\psi} &= \frac{v}{l}\sin\beta \end{aligned} \tag{2}$$

where (x,y) is the position of the vehicle, v its forward speed, ψ its heading, a the acceleration command, and β the slip angle (i.e., the angle between the direction of the vehicle and the direction of the front wheels) at the center of gravity, defined in Eq. (3):

$$\beta = \tan^{-1}(1/2\tan\ \delta) \tag{3}$$

where δ is the front wheel angle, which is used as the steering command.

We define a simulation step as a time interval in which the environment state and the derived agent observations are updated according to the changes in the scene. The frequency of the

simulation steps is set to 5 Hz. A training step is composed of multiple simulation steps after which the agent's behavior is updated according to the state-action-reward tuples for each simulation step in the algorithms replay buffer. All the variables of the kinematic model are calculated at each model simulation step, and such calculations allow the propagation of the vehicle state.

4.4 Reward

We define the RF for this problem as the weighted sum of 3 rewards that are provided by the environment:

$$R = c_{coll} \cdot R_{coll} + c_{rml} \cdot R_{rml} + c_{hs} \cdot R_{hs} \tag{4}$$

where:

- R_{coll} (Eq. 5) is the collision reward and is equal to -1 when a collision happens. In that case, the training episode is also terminated, in order to prevent the agent to accumulate positive rewards. On the other hand, if the EV reaches the end of the simulation without crashing, then R_{coll} is equal to 0.

$$R_{coll} = \begin{cases} -1 \; if \; the \; vehicle \; crashes \\ 0 \quad otherwise \end{cases} \tag{5}$$

- R_{rml} (Eq. 6) is the right-most lane reward, which is employed to encourage the EV to keep the right-most lane whenever it is possible. The maximum reward is given when the RV travels on the right-most lane. The reward linearly decreases until the left-most lane, where it drops to 0. In (Eq. 6), N is the number of lanes in the road. Differently from the previous one, this reward is dense, meaning that it is given to the learning agent at each simulation step.

$$\mathrm{R}_{rml} = \frac{current \; EV \; lane \; index}{N} \tag{6}$$

- R_{hs} (Eq. 7) is the high-speed reward and is designed to encourage the EV to keep high traveling speed values. This reward is dense, as well. The maximum reward is attributed when the cruising speed of the EV is between a certain interval, which in our case goes from roughly 30 to 36 m/s (108 to 130 km/h).

$$R_{hs} = clip(scaled \; speed, 0, 1) \tag{7}$$

  $scaled \; speed$ is a linear interpolation described as:

$$scaled \; speed = \left(\frac{v - l_{min}}{l_{max} - l_{min}} \right) \tag{8}$$

  where l_{min} and l_{max} are respectively the extremes of a speed range in which the agent receives a positive reward and v its forward speed.

The output value of dense rewards is linearly mapped between 0 and 0.1 before being fed to the DRL algorithm, while the collision reward is added as is. This normalization process is necessary to avoid a significant increase in the numerical value of the total reward and, therefore, to stabilize the learning phase.

In the next section, we analyze how, by changing the weights of the proposed reward terms, it is possible to model three different driving styles, such as:

- Standard: normal behavior adopted when driving on a highway. It should involve a limited number of collisions.

- Comfort: this driving style shows a preference to occupy the rightmost lane, and travel at a relatively slow speed.
- Aggressive: behavior that implies high traveling speed, a high number of overtaking maneuvers, and quick decision changes.

5. Experimental results

This section presents the results obtained by testing the above described design. Results refer to a training of the agents performed on a laptop with an Intel Core i7-12700H CPU, 16 GB of RAM, and an NVIDIA RTX 3060 GPU.

For the DRL training, we employed a PPO algorithm and tuned the hyperparameters by setting the number of policy update steps to 4096, batch size to 128, gamma to 0.997, and entropy coefficient to 0.1. In PPO, a policy update step occurs after a specified number of simulation steps (4096 in our case), such as for the mentioned training step, and involves selecting a new agent policy, and consequently updating the policy gradient, on the basis of the state-action-reward-new state tuples observed since the last policy update step. The gamma hyperparameter refers to the algorithm's discount factor used to balance the newly obtained rewards in the agent's objective function. In this context, the entropy coefficient quantifies the degree of uncertainty the agent has about its actions given a particular state. In PPO, it is used to encourage the model to explore the environment by penalizing excessively deterministic and fixed behaviors on which the agent may already have converged.

The DNN configuration is the highway-env default 2-layer multi-layer perceptron (MLP), with 64 neurons per layer. We used a 3-lane highway environment (Figure 2), setting policy frequency (i.e., decision rate) to 1 Hz, in order to give the agent sufficient time to observe the complete effects of its previous decision. The training procedure consists of about 5000 episodes, each one of a maximum length of 60 seconds, and encompassing about 200k training updates. The whole training is completed within 90 minutes.

5.1 High-level Decision Making

Figure 5a shows the evolution of the training of a standard agent in a Tensorboard view [52]. The standard agent has the following weights for the RF terms: -3 for collisions, 0.4 for speed, and 0.2 for the right lane (negative weights indicate penalties). The average episode length reaches its maximum (i.e., 60 seconds, meaning no collisions occurred) at around 60K iterations. The average total reward (Figure 5b) has a similar shape since it is strongly influenced by the collision penalty. We notice that the training is quite rapid, smooth, and stable, without catastrophic forgetting [53], which frequently affects the DRL training.

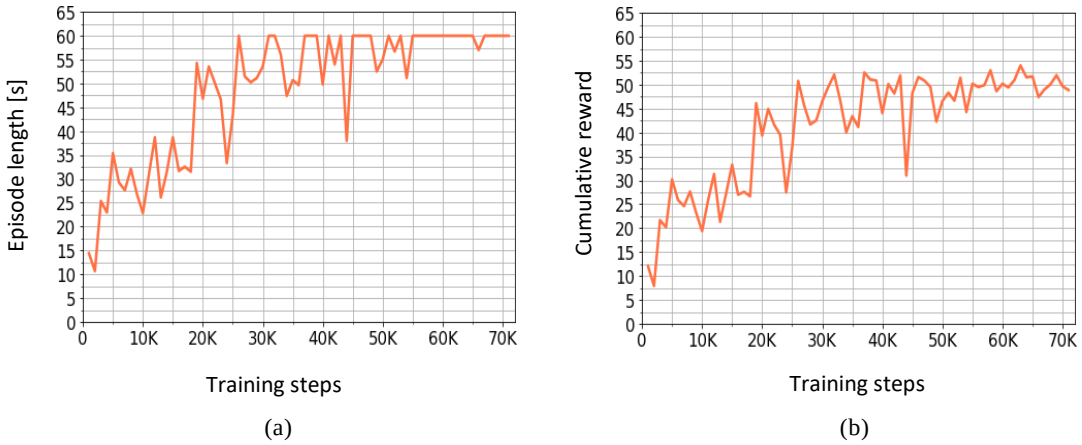

Figure 5. Training metrics (a) episode length and (b) total reward as a function of number of iterations.

Table 1. Performance comparison over sixty 60-second episodes

Model	Collisions per episode	Km per episode
High-level DM agent	0	1.83
Low-level agent [15]	0.03	1.76

Table 2. Environment settings

Model	Speed levels [m/s]	No. vehicles per episode	Traffic density	No. reward functions
High-level DM agent	[20, 25, 30]	10	0.5	3
Low-level agent [15]	Cont(20, 36)	Uniform(10-20)	0.4	11

It appears that the high-level agent performs better under both the considered dimensions. It is important to highlight that the training of the higher-level agent took more than one order of magnitude less training steps than the lower-level agent (70K vs. 1.75 M). We argue that such a significant reduction is motivated by the higher simplicity of the action space in the higher-level decision case, in which the agent can exploit state-of-the-art ADASs for the lower-level longitudinal and lateral motion control, and thus focus on the strategic decisions only. An expert assessment also noticed a more realistic behavior by the high-level agent [54]. A quantitative indicator for this is given by the elimination of the collisions.

Concerning the proposed comparison, it must be specified that the two environments are very similar to each other, but not exactly the same. Table 2 shows that an episode in Campodonico et al. [15] has fewer total vehicles (EV + NPVs), but they are slightly more concentrated (density factor 0.5 vs 0.4). The model presented in [15] was specifically developed with a large number of rewards compared to the original highway-env (11 vs 3), in order to better enforce traffic law and safer behavior (e.g., penalties for right overtake, unsafe distance to the front vehicle, hazardous lane change, steering angle).

5.2 Driving styles

The reduction in training time ensured by the higher-level DM module provides a significant advantage for training more agents, each one featuring a specific driving style. We explored this by specifying different weights for the rewards and penalties with which the agent is trained, as shown in Table 3. In all cases, the training episode was interrupted at the occurrence of a collision, thus preventing the agent to get any further reward for that episode.

Table 3. Reward weights for training three different driving styles.

Driving style	Collision	Speed	Right lane
Comfort	30	0	0.8
Standard	3	0.4	0.2
Aggressive	0	0.8	0

The training of the comfort driving style is characterized by a high penalty to prevent collisions and a high reward to encourage the EV to drive in the rightmost lane. On the other hand, the aggressive behavior's training provides no collision penalty and a high-speed reward. Finally, the standard driving style is trained as described in the previous sub-section.

Figure 6 shows the evolution during the training of four different quantities: episode length, high-speed reward, right-most lane reward and collision reward (actually, penalty). Zero-weighted rewards are ignored.

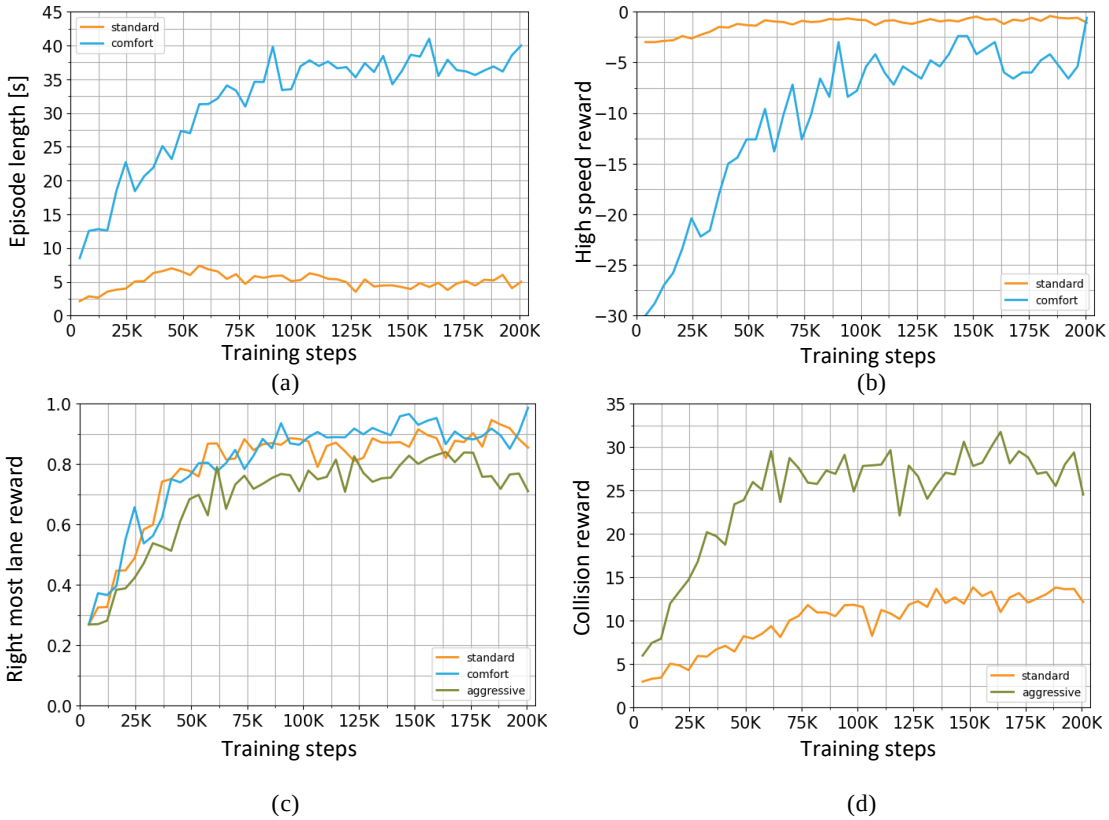

Figure 6. Evolution of the training for the different driving styles. (a) Episode length, normalized from 0 to 1 (i.e., 60 seconds), (b) High speed reward (not assigned to the comfort model), (c) Right-most lane reward (not assigned to the aggressive model) and (d) collision reward (not assigned to the aggressive model)

Figure 6a shows that all the three models tend to achieve a similar value of episode length, which indicates the overall importance of completing each episode (i.e., avoid collisions). However, Figure 6b, 6c, and 6d indicate that this is achieved through different types of behavior. Figure 6b shows that the high-speed reward gained by the aggressive agent is much higher than for the standard agent, since its weight is twice as much (0.8 and 0.4, respectively-Table 3). Even more apparent, the right-most lane reward for the comfort driving agent increases significantly, differently from the standard model, as its weight is four times larger (0.8 vs. 0.2) (Figure 6). Figure 6d, on the other hand, is significant, since it indicates that designing the reward factors may be tricky. Particularly, we see that, while the training of the standard agent is smooth, the training of the comfort agent is more complex, since the huge collision penalty puts a burden that lasts for a long time. This spurred us to consider an optimization to the training process, that we will present later in this section. It is important to highlight that the episode length results in Table 4 differ a bit from those in Figure 6a because they are obtained in exploitation. In contrast, Figure 6a refers to training, which also includes a fraction of environment exploration, leading to lower results.

Table 4 presents some typical vehicular metrics that we analyze in order to check whether the different training actually led to their respectively targeted driving styles. Particularly, it is apparent that the comfort model tends to drive in the right-most lane, never overtaking, at a much lower speed than the other driving styles. Moreover, considering the number of decision changes, the comfort model seems to follow a stable policy. On the other hand, the aggressive model features a much higher collision rate, average speed and decision change rate. Overall, these two models look less balanced than the standard one, which drives at high speed, with few collisions.

Until now, we have trained each one of the three different agents from scratch. But this has involved some inconvenience, as shown in Figure 6d. In this last part of the section we explore an alternative technique, called curriculum learning (CL) [55], which consists of training a ML model starting with easier samples and then gradually increase their difficulty, as the model learns. Thus, we tried to train the comfort and aggressive models (i.e., using the specific weights of their RF), starting their learning phase from a standard model (i.e., the weights of the deep neural networks of the comfort and aggressive agents are initialized with the values of the weights of a fully trained standard model), thus embodying its previous knowledge.

We explored CL applying the differentiated RFs for half the number of training steps (100k, compared to the 200k employed for training from scratch). Table 5 shows that the CL models are more closely related to the standard driving policy than those trained with the classical approach. For instance, a CL aggressive agent features a much lower collision rate and higher mileage. Similarly, the collision rate and decision change rate of the CL-trained comfort model are quite higher. These differences highlight that the CL strategies are learned atop of the baseline model, while the "from scratch" trained agents build up their experience during training without any prior influence.

Table 4. Performance over 1000 episodes of the different driving styles obtained from scratch.

Model's style	Avg. ep. length [s]	Collision rate	Avg. km travelled	Avg. speed [m/s]	Avg. decel. [m/s^2]	Avg. acc. [m/s^2]	Avg. dec. change	Avg. left change	Avg. right change
Comfort	60	0.5%	1.43	**23.9**	**-2.76**	**2.46**	**3.61**	0.01	**0.32**
Standard	59	1.2%	**1.82**	30.6	-4.48	2.29	8.22	0.52	0.74
Aggressive	49	**29.2%**	1.62	32.8	**-4.89**	**2.15**	12.46	**1.68**	1.69

Table 5. Performance over 1000 episodes of the different driving styles obtained from curriculum learning.

Model's style	Avg. ep. length [s]	Collision rate	Avg. km travelled	Avg. speed [m/s]	Avg. decel. [m/s^2]	Avg. acc. [m/s^2]	Avg. dec. change	Avg. left change	Avg. right change
Comfort	60	1%	1.43	**23.9**	-2.88	2.59	6.22	**0**	0.45
Standard	59	1.2%	1.82	30.6	-4.48	2.29	8.22	0.52	0.74
Aggressive	**55**	**11.9%**	**1.92**	33.3	-4.76	1.83	**13.15**	1.14	**0.96**

6. Conclusions and future work

The paper has explored the training of DRL models deployable as realistic NPVs in driving SGs. We have introduced a hierarchical architecture for behavioral planning of vehicle models, in which the agents decide their motion by taking "human-like", high-level decisions, such as

"keep lane", "overtake," and "go to rightmost lane". This is similar to a driver's high-level reasoning and takes into account the availability of ever more sophisticated ADAS in current vehicles. Compared to a low-level DM system, our model performs better both in terms of safety and speed. As a significant advantage, the proposed approach allows reducing the number of training steps by more than one order of magnitude. This makes the development of new models much more efficient, which is key for implementing different vehicles, featuring different driving styles.

As a second main contribution to advance the state of the art, we demonstrated that it is possible to train agents characterized by different driving behaviors, by tweaking the weights of the factors of a general RF. This approach avoids heuristic coding of the driving styles, saving development and maintenance time. The proposed agent models can then be seamlessly deployed in any target SG in the same environment as the one we used for training (i.e., highway-env). Furthermore, the information and lessons learned presented in this article can be useful to train similar agents in their target SG environment.

In order to optimize the training, we employed the curriculum learning technique, starting the training procedure of a more specialized agent from a base model rather than from scratch. Results indicate that curriculum learning allows achieving the same performance, but in much less training iterations (thus time). However, the specialized agents tend to significantly "inherit" behavior from the baseline agent, which may not always be desirable.

This paper concerns development of NPVs for driving SGs, in contexts like driving instruction [2] and safety awareness [3]. However, the key concepts of our work (i.e., training autonomous agents for various types of a SG non-player characters, and hierarchical modeling) are general, and may be adapted to other SGs and simulation domains as well.

We argue that future work could explore two main directions to increase the realism of the vehicle behavior: the use of more accurate dynamic models inside highway-env, overcoming the limits of the kinematic model, and the improvement of the robustness of the training paradigm through a multi-agent adversarial policy (e.g., [56]). Finally, after having verified technical feasibility, a fundamental next step will consist in assessing the effect of the developed models on player's learning and engagement in a SG, in a duly designed user study. Another addition to the work, is to test the presented ideas in other contexts, allowing SGs developers to handle the implementation of more realistic SGs features improving the realism and user experience.

Acknowledgments

The authors would like to thank Nicola Poerio, of CRF Stellantis, for his invaluable support on the subject of reinforcement learning for automated driving.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] "The Good Drive, a serious game for learning to drive - Renault Group." Accessed: Jul. 22, 2022. [Online]. Available: https://www.renaultgroup.com/en/news-on-air/news/the-good-drive-a-serious-game-for-learning-to-drive/

[2] P. Backlund, H. Engström, M. Johannesson, and M. Lebram, "Games for traffic education: An experimental study of a game-based driving simulator," *Simulation & Gaming*, vol. 41, no. 2, pp. 145–169, Apr. 2010, doi: 10.1177/1046878107311455.

[3] A. Gounaridou, E. Siamtanidou, and C. Dimoulas, "A Serious Game for Mediated Education on Traffic Behavior and Safety Awareness," *Education Sciences*, vol. 11, no. 3, Art. no. 3, Mar. 2021, doi: 10.3390/educsci11030127.

[4] P. Wouters and H. Van Oostendorp, "Overview of Instructional Techniques to Facilitate Learning and Motivation of Serious Games," in *Instructional Techniques to Facilitate Learning and Motivation of Serious Games*, P. Wouters and H. Van Oostendorp, Eds., Cham: Springer International Publishing, 2017, pp. 1–16. doi: 10.1007/978-3-319-39298-1_1.

[5] F. Tena-Chollet, J. Tixier, A. Dandrieux, and P. Slangen, "Training decision-makers: Existing strategies for natural and technological crisis management and specifications of an improved simulation-based tool," *Safety Science*, vol. 97, pp. 144–153, Aug. 2017, doi: 10.1016/j.ssci.2016.03.025.

[6] E. Leurent and J. Mercat, "Social Attention for Autonomous Decision-Making in Dense Traffic." arXiv, Nov. 27, 2019. doi: 10.48550/arXiv.1911.12250.

[7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016, doi: 10.1109/TITS.2015.2498841.

[8] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," *FNT in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, 2018, doi: 10.1561/2200000071.

[9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[10] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep Reinforcement Learning on Autonomous Driving Policy With Auxiliary Critic Network," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2023, doi: 10.1109/TNNLS.2021.3116063.

[11] F. Bellotti, L. Lazzaroni, A. Capello, M. Cossu, A. De Gloria, and R. Berta, "Explaining a Deep Reinforcement Learning (DRL)-Based Automated Driving Agent in Highway Simulations," *IEEE Access*, vol. 11, pp. 28522–28550, 2023, doi: 10.1109/ACCESS.2023.3259544.

[12] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep Reinforcement Learning on Autonomous Driving Policy With Auxiliary Critic Network," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2023, doi: 10.1109/TNNLS.2021.3116063.

[13] A. J. M. Muzahid, S. F. Kamarulzaman, Md. A. Rahman, and A. H. Alenezi, "Deep Reinforcement Learning-Based Driving Strategy for Avoidance of Chain Collisions and Its Safety Efficiency Analysis in Autonomous Vehicles," *IEEE Access*, vol. 10, pp. 43303–43319, 2022, doi: 10.1109/ACCESS.2022.3167812.

[14] F. Bellotti, L. Lazzaroni, A. Capello, M. Cossu, A. De Gloria, and R. Berta, "Explaining a Deep Reinforcement Learning (DRL)-Based Automated Driving Agent in Highway Simulations," *IEEE Access*, vol. 11, pp. 28522–28550, 2023, doi: 10.1109/ACCESS.2023.3259544.

[15] G. Campodonico *et al.*, "Adapting Autonomous Agents for Automotive Driving Games," in *Games and Learning Alliance*, F. de Rosa, I. Marfisi Schottman, J. Baalsrud Hauge, F. Bellotti, P. Dondio, and M. Romero, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 101–110. doi: 10.1007/978-3-030-92182-8_10.

[16] "GitHub - eleurent/highway-env: A minimalist environment for decision-making in autonomous driving." Accessed: Jul. 11, 2022. [Online]. Available: https://github.com/eleurent/highway-env

[17] E. Leurent, "An Environment for Autonomous Driving Decision-Making." May 2018. Accessed: Apr. 17, 2023. [Online]. Available: https://github.com/eleurent/highway-env

[18] G. Brockman *et al.*, "OpenAI Gym," no. arXiv:1606.01540. arXiv, Jun. 05, 2016. Accessed: Apr. 17, 2023. [Online]. Available: http://arxiv.org/abs/1606.01540

[19] U. T. SL, "Car Driving School Simulator (Android)," Uptodown.com. Accessed: Jul. 22, 2022. [Online]. Available: https://car-driving-school-simulator.en.uptodown.com/android

[20] K. Likitweerawong and P. Palee, “The virtual reality serious game for learning driving skills before taking practical test,” *2018 International Conference on Digital Arts, Media and Technology (ICDAMT)*, 2018, doi: 10.1109/ICDAMT.2018.8376515.
[21] H. Hrimech *et al.*, “The Effects of the Use of Serious Game in Eco-Driving Training,” *Frontiers in ICT*, vol. 3, 2016, Accessed: Jul. 22, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fict.2016.00022
[22] K. Cho, W. Song, and K. Um, “Gaussian Distribution for NPC Character in Real-Life Simulation,” in *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, Oct. 2007, pp. 132–135. doi: 10.1109/IPC.2007.90.
[23] M. D. Kickmeier-Rust, S. Göbel, and D. Albert, “80Days: Melding adaptive educational technology and adaptive and interactive storytelling in digital educational games,” in *Proceedings of the First International Workshop on Story-Telling and Educational Games (STEG’08)*, Citeseer, 2008.
[24] M. Zielke *et al.*, “Serious Games for Immersive Cultural Training: Creating a Living World,” *IEEE computer graphics and applications*, vol. 29, pp. 49–60, May 2009, doi: 10.1109/MCG.2009.30.
[25] R. Massoud, F. Bellotti, R. Berta, A. De Gloria, and S. Poslad, “Eco-driving Profiling and Behavioral Shifts Using IoT Vehicular Sensors Combined with Serious Games,” in *2019 IEEE Conference on Games (CoG)*, London, United Kingdom: IEEE, Aug. 2019, pp. 1–8. doi: 10.1109/CIG.2019.8847992.
[26] F. Sagberg, Selpi, G. F. Bianchi Piccinini, and J. Engström, “A Review of Research on Driving Styles and Road Safety,” *Hum Factors*, vol. 57, no. 7, pp. 1248–1275, Nov. 2015, doi: 10.1177/0018720815591313.
[27] F. Bellotti, R. Berta, and A. De Gloria, “A Social Serious Game Concept for Green, Fluid and Collaborative Driving,” in *Applications in Electronics Pervading Industry, Environment and Society*, vol. 289, A. De Gloria, Ed., in Lecture Notes in Electrical Engineering, vol. 289. , Cham: Springer International Publishing, 2014, pp. 163–170. doi: 10.1007/978-3-319-04370-8_15.
[28] D. Troullinos, G. Chalkiadakis, D. Manolis, I. Papamichail, and M. Papageorgiou, “Extending SUMO for Lane-Free Microscopic Simulation of Connected and Automated Vehicles,” *SUMO Conf Proc*, vol. 3, pp. 95–103, Sep. 2022, doi: 10.52825/scp.v3i.110.
[29] P. A. Lopez *et al.*, “Microscopic Traffic Simulation using SUMO,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI: IEEE, Nov. 2018, pp. 2575–2582. doi: 10.1109/ITSC.2018.8569938.
[30] R. Massoud, S. Poslad, F. Bellotti, R. Berta, K. Mehran, and A. D. Gloria, “A Fuzzy Logic Module to Estimate a Driver’s Fuel Consumption for Reality-Enhanced Serious Games,” *International Journal of Serious Games*, vol. 5, no. 4, Art. no. 4, Dec. 2018, doi: 10.17083/ijsg.v5i4.266.
[31] L. Lazzaroni, A. Mazzara, F. Bellotti, A. De Gloria, and R. Berta, “Employing an IoT Framework as a Generic Serious Games Analytics Engine,” in *Games and Learning Alliance*, I. Marfisi-Schottman, F. Bellotti, L. Hamon, and R. Klemke, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 79–88. doi: 10.1007/978-3-030-63464-3_8.
[32] W. Westera *et al.*, “Artificial intelligence moving serious gaming: Presenting reusable game AI components,” *Education and Information Technologies*, vol. 25, no. 1, pp. 351–380, Jan. 2020, doi: 10.1007/s10639-019-09968-2.
[33] S. L. Tomlinson and N. Melder, “An Architecture Overview for AI in Racing Games,” 2015. doi: 10.1201/b16725-44.
[34] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, “End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017, pp. 474–475. doi: 10.1109/CVPRW.2017.64.

[35] A. Fakhry, “Applying a Deep Q Network for OpenAI’s Car Racing Game,” Medium. Accessed: Jul. 22, 2022. [Online]. Available: https://towardsdatascience.com/applying-a-deep-q-network-for-openais-car-racing-game-a642daf58fc9
[36] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning.” arXiv, Dec. 19, 2013. doi: 10.48550/arXiv.1312.5602.
[37] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach Learn*, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.
[38] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” in *Advances in Neural Information Processing Systems*, MIT Press, 1999. Accessed: Jul. 22, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html
[39] T. Wang *et al.*, “Benchmarking Model-Based Reinforcement Learning.” arXiv, Jul. 03, 2019. doi: 10.48550/arXiv.1907.02057.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms.” arXiv, Aug. 28, 2017. doi: 10.48550/arXiv.1707.06347.
[41] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust Region Policy Optimization.” arXiv, Apr. 20, 2017. Accessed: Apr. 30, 2023. [Online]. Available: http://arxiv.org/abs/1502.05477
[42] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning.” arXiv, Jun. 16, 2016. doi: 10.48550/arXiv.1602.01783.
[43] A. Capello *et al.*, “Investigating High-Level Decision Making for Automated Driving,” in *Applications in Electronics Pervading Industry, Environment and Society*, R. Berta and A. De Gloria, Eds., in Lecture Notes in Electrical Engineering. Cham: Springer Nature Switzerland, 2023, pp. 307–311. doi: 10.1007/978-3-031-30333-3_41.
[44] M. Pleines *et al.*, “On the Verge of Solving Rocket League using Deep Reinforcement Learning and Sim-to-sim Transfer,” in *2022 IEEE Conference on Games (CoG)*, Beijing, China: IEEE, Aug. 2022, pp. 253–260. doi: 10.1109/CoG51982.2022.9893628.
[45] L. Lazzaroni, F. Bellotti, A. Capello, M. Cossu, A. De Gloria, and R. Berta, “Deep Reinforcement Learning for Automated Car Parking,” in *Applications in Electronics Pervading Industry, Environment and Society*, R. Berta and A. De Gloria, Eds., in Lecture Notes in Electrical Engineering. Cham: Springer Nature Switzerland, 2023, pp. 125–130. doi: 10.1007/978-3-031-30333-3_16.
[46] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-Baselines3: Reliable Reinforcement Learning Implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
[47] E. Leurent and J. Mercat, “Social Attention for Autonomous Decision-Making in Dense Traffic.” arXiv, Nov. 27, 2019. Accessed: May 02, 2023. [Online]. Available: http://arxiv.org/abs/1911.12250
[48] A. Kesting, M. Treiber, and D. Helbing, “General Lane-Changing Model MOBIL for Car-Following Models,” *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, Jan. 2007, doi: 10.3141/1999-10.
[49] G. Brockman *et al.*, “OpenAI Gym.” arXiv, Jun. 05, 2016. Accessed: Apr. 17, 2023. [Online]. Available: http://arxiv.org/abs/1606.01540
[50] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, “High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI: IEEE, Nov. 2018, pp. 2156–2162. doi: 10.1109/ITSC.2018.8569448.
[51] P. Polack, F. Altché, B. d’Andréa-Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 812–818. doi: 10.1109/IVS.2017.7995816.

[52] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” arXiv, Mar. 16, 2016. doi: 10.48550/arXiv.1603.04467.
[53] J. Kirkpatrick *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, doi: 10.1073/pnas.1611835114.
[54] L. Forneris, “Offline Reinforcement Learning for Autonomous Driving Following an Actor-Critic Approach, M.S. thesis, Università degli Studi di Genova, Genova, Oct. 2022,” M.S. thesis, Università degli Studi di Genova, Genova.
[55] X. Wang, Y. Chen, and W. Zhu, “A Survey on Curriculum Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3069908.
[56] A. Sharif and D. Marijan, “Adversarial Deep Reinforcement Learning for Improving the Robustness of Multi-agent Autonomous Driving Policies,” in *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2022, pp. 61–70. doi: 10.1109/APSEC57359.2022.00018.