Knowledge-Based Systems 293 (2024) 111663

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Dealing with uncertainty: Balancing exploration and exploitation in deep recurrent reinforcement learning

Valentina Zangirolami, Matteo Borrotti [*]

Department of Economics, Management and Statistics, University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, Milano, 20126, Italy

ARTICLE INFO

Dataset link: https://github.com/ValentinaZangirolami/DRL

MSC:
0000
1111

Keywords:
Exploration strategies
Deep recurrent reinforcement learning
Autonomous driving

ABSTRACT

Incomplete knowledge of the environment leads an agent to make decisions under uncertainty. One of the major dilemmas in Reinforcement Learning (RL) where an autonomous agent has to balance two contrasting needs in making its decisions is: exploiting the current knowledge of the environment to maximize the cumulative reward as well as exploring actions that allow improving the knowledge of the environment, hopefully leading to higher reward values (exploration–exploitation trade-off). Concurrently, another relevant issue regards the full observability of the states, which may not be assumed in all applications. For instance, when 2D images are considered as input in an RL approach used for finding the best actions within a 3D simulation environment. In this work, we address these issues by deploying and testing several techniques to balance exploration and exploitation trade-off on partially observable systems for predicting steering wheels in autonomous driving scenarios. More precisely, the final aim is to investigate the effects of using both adaptive and deterministic exploration strategies coupled with a Deep Recurrent Q-Network. Additionally, we adapted and evaluated the impact of a modified quadratic loss function to improve the learning phase of the underlying Convolutional Recurrent Neural Network. We show that adaptive methods better approximate the trade-off between exploration and exploitation and, in general, Softmax and Max-Boltzmann strategies outperform ϵ-greedy techniques.

1. Introduction

Reinforcement learning (RL) is a core topic in machine learning and is concerned with sequential decision-making in an uncertain environment. Two key concepts in RL are exploration, which consists of learning via interactions with an unknown environment, and exploitation, which consists of optimizing the objective function given accumulated information. In such a scenario, an RL algorithm repeatedly makes decisions to maximize its rewards, the so-called exploitation; the RL algorithm, however, has only limited knowledge about the process of generating the rewards. Thus, occasionally, the algorithm might decide to perform exploration which improves the knowledge about the reward generating process, but which is not necessarily maximizing the current reward [1]. Exploitation can be studied using the (stochastic) control theory, while exploration relies on the theory of statistical learning. These two concepts are complementary but opposite: exploration leads to the maximization of the gain in the long run at the risk of losing short-term reward, while exploitation maximizes the short-term gain at the price of losing the gain over the long run. A careful trade-off between these two objectives is important to the success of any learner.

The main motivation of this work is to address the issues of partial observability of states together with the exploration–exploitation trade-off through deterministic and stochastic strategies. The final aim is to provide a comprehensive analysis of reinforcement learning, with a focus on deep recurrent reinforcement learning from the point of view of the previously mentioned trade-off. In summary, the main contributions of this work can be summarized as follows:

(1) Refine the quadratic loss function proposed by Lample and Chaplot [2] to align with the Bootstrapped Random Update sampling technique. While Lample and Chaplot [2] originally developed this strategy for Bootstrapped Sequential Update sampling technique. The introduction of this strategy to deep recurrent reinforcement learning leads to a speed-up of the algorithm convergence.
(2) Conduct a comprehensive deep analysis and comparison of several exploration strategies on partially observable systems. Specifically, we evaluated several ϵ-greedy and Softmax approaches, incorporating Bayesian perspectives and assessing the agent's uncertainty to adjust the ϵ probability dynamically. Originally,

[*] Corresponding author.
E-mail address: matteo.borrotti@unimib.it (M. Borrotti).

https://doi.org/10.1016/j.knosys.2024.111663
Received 9 June 2023; Received in revised form 8 March 2024; Accepted 17 March 2024
Available online 19 March 2024

these methods were proposed by Tokic [3], Tokic and Palm [4], and Gimelfarb et al. [5] for fully observable system states, we successfully adapted all strategies to the recurrent framework.

(3) Modify the Value Difference Based Exploration (VDBE) method to suit its adaptation and integration into the Deep Recurrent Q-learning and, more broadly, Deep Q-Learning scenarios. Indeed, we consider the difference among estimated Q-values with different estimated weights of neural networks (related to previous and current updates of the agent).

(4) Implement a simulation study to predict the steering wheel angle in autonomous driving systems. Within this application, we examined the impact of the exploration strategies in partially observable systems for balancing exploration. To address the challenges posed by partially observable states, we proposed Deep Double Dueling Recurrent Q-Learning with a modified loss to enhance collision avoidance performance while keeping the speed constant, evaluating solely the vehicle's steering control.

This paper is organized as follows. Section 2 describes related work based on exploration strategies, recurrent models of DRL and autonomous driving. Section 3 provides a comprehensive overview of reinforcement learning, highlighting key issues and introducing deep recurrent reinforcement learning. Section 4 delves into the details of Deep Recurrent Q-learning model tailored for self-driving cars specifying our loss proposal for Bootstrapped Random updates. Section 5 elucidates the exploration strategies compared in the experiments emphasizing their theoretical aspects within a partial observable framework. Section 6 outlines the experimental settings, encompassing the use of AirSim simulator, the parameters of exploration strategies, and DRL model. Additionally, it shows the results obtained from the training and test set, along with a deep analysis of the impacts of exploration strategies. Section 7 concludes this paper with final considerations and outlines for future works.

2. Related literature

This work is related to three major research fields, namely exploration strategies, function gradient approximation for RL and autonomous driving.

2.1. Exploration strategies

In the literature, many different exploration methods for RL processes are based on a discrete action space, in which Softmax and ϵ-greedy are the most popular. Ortiz et al. [6] considered a deterministic ϵ-greedy strategy with linear decreasing using four different models, including Q-learning. This kind of ϵ-greedy method is significantly simplified, where ϵ is the inverse of the step number without defining the lower bound to be reached in the final step. Gimelfarb et al. [5] and Tokic [3] proposed two novel adaptive methods for ϵ-greedy on full-observable domain and used training information to update ϵ probability. Tokic [3] compared constant ϵ-greedy and softmax with an adaptive ϵ-greedy strategy based on Q-value differences as a measure of uncertainty. Gimelfarb et al. [5] used Bayesian Inference to estimate the ϵ value with theoretical convergence guarantee and compare it with deterministic and adaptive ϵ-greedy approaches. However, both papers conducted a limited comparison of strategies. Gimelfarb et al. [5] compare only ϵ-greedy strategies while Tokic [3] only includes constant ϵ-greedy while Softmax. Tokic and Palm [4] proposed an extension of Max-Boltzmann Exploration in a full-observable states domain which combines Softmax and value-based adaptive ϵ-greedy. Tokic and Palm [4] compared this novel strategy with constant ϵ-greedy, VDBE, and Softmax regardless of decreasing ϵ-greedy and other adaptive strategies. Cruz et al. [7] studied the effect of many exploration strategies. Among others, the baseline form of ϵ-greedy and Softmax with the extensions proposed by Tokic and Palm [4] and Tokic [3]. Our work extends the previous analysis by investigating the impact of other techniques such as Max-Boltzmann Exploration and Decreasing ϵ-greedy strategy, where the decreasing linear equations assume changes in slope depending on the progress in the training learning. Furthermore, we analyzed all these methods in order to evaluate the balance of exploration–exploitation trade-off on a partially observable system.

2.2. Function gradient-based RL

The success of RL in decision-making has recently enticed researchers to apply Deep Q-Learning (DQL) methods on video games and autonomous driving tasks, which all require Convolutional Neural networks to approximate value functions with image-based states. In literature, there are several models used to estimate Q-values in discrete control systems which could be based on Full and Partial observability of the states. Minwoo et al. [8] suggested the extension of Deep Q-Network (DQN) approach, like Double Deep Q-Network (DDQN) and Double Dueling Deep Q-Network (D3QN), to find the optimal action avoiding obstacles on autonomous drone. Minwoo et al. [8] showed that DDQN and D3QN overcome the performance of the baseline DQN structure, in which D3QN learns better policies than other methods. Several works [9–13] evaluated Deep Q-Learning in several RL applications. However, DQN and its extensions assume the full observability of the states that fall in the real world and, in general, 3D environment. The real-world environment is characterized by uncertainty whereas a system based on full observability fails to capture the true dynamics as there might be noisy sensors, missing information about the state, or outside interferences. Hausknecht and Stone [14] proposed two main sampling methods of Deep Recurrent Q-Learning (DRQL) and they used Bootstrapped Random Updates from memory replay for Neural Network weight optimization. Moreover, Hausknecht and Stone [14] compared DQN and Deep Recurrent Q-Network (DRQN) methods on two game environments, where DRQN performed both well and poorly. In fact, Hausknecht and Stone [14] showed how DRQN updates might trigger some problems with the learning of functions which could be reflected in the final performance. Other papers [15–18] compared Deep Q-Network and Deep Recurrent Q-Network in different fields to evaluate the performance on a partially observable domain. However, most of these studies did not achieve good results when systems were based on hidden states. Lample and Chaplot [2] proposed a novel method for Deep Recurrent Q-Network so as to overcome challenges during the agent's update. Consistently, Lample and Chaplot [2] proposed a technique based on an error mask in the optimization phase in order to update neural network weights with enough history of observations. Lample and Chaplot [2] tested their techniques coupled with Bootstrapped Sequential Update as a sampling strategy. In this work, we extend the contribution of Lample and Chaplot [2] to Bootstrapped Random Update sampling strategy.

2.3. Autonomous driving

The use of Convolutional Neural Network (CNN) in Autonomous Driving has been a topic of great interest in recent years. Some works have applied Deep Reinforcement Learning in order to automate vehicle and process images with more scalability. Several researchers studied autonomous driving tasks by using simulators that generate iterative images and perform actions in real time. Wu et al. [19] and Santara et al. [20] considered TORCS simulator platform to build DRL frameworks. Generally, TORCS is used to simulate a car racing environment, which is useful for managing and training the agent in situations where other cars are present. However, a car racing environment does not include several real-world peculiarities like parked cars, static obstacles, animals, and vegetation. Xiao et al. [21] and Michelmore et al. [22] show the results obtained by two different DRL frameworks using CARLA simulator. In comparison to TORCS, CARLA provides environments that are visually closer to reality. AirSim and

CARLA are very similar, which is the reason why Pilz et al. [23] assigned positive ratings to these two simulators in terms of interface compatibility, access to ego vehicle data, access to nonego vehicle data, access to pedestrian data, detail and variety of sensors, detail of the rendered graphics, detail of the physics engine and cost efficiency. In the literature, there are some works on DRL for autonomous driving. Riboni et al. [24] compared DDQN and D3QN models with transfer learning techniques via decreasing ϵ-greedy in order to evaluate collision avoidance performance assuming a discrete action space of the steering angle. Deshpande et al. [25] proposed a DRQN model with descending ϵ-greedy to control car's steering angle and speed, as Liao et al. [26] compared DQN and DDQN models for the same objective. Our study aims to overcome these papers by considering several exploration strategies on a partially observable system, which proves a central issue in dealing with 3D environments. Moreover, previous works only considered deterministic strategies for balancing exploration, which may not have been efficient in approximating agent uncertainty because of a ϵ exogenous.

3. Background and preliminaries

In this section we delve into the concept of *exploration* within reinforcement learning, presenting prior frameworks addressing similar challenges. We subsequently introduce fundamental notions and theoretical background of reinforcement learning, emphasizing its application in partially observable environments and exploring the realm of deep recurrent reinforcement learning.

3.1. Multi-armed bandits and contextual bandits

The multi-armed bandit problem (Rodman [27], Whittle [28], Bubeck and Cesa-Bianchi [29]) is the simplest framework that encompasses the key issue of *balancing exploration and exploitation*. In the aim of maximizing the cumulative discounted future rewards, the current decision about the action to be taken revolves around a dilemma: yielding the highest reward (*exploitation*) or exploring new actions (*exploration*) thereby gaining additional knowledge for improving future decisions.

The stochastic multi-armed bandit problem could be represented by a set of actions $\mathcal{A} = \{1, \dots, K\}$, an agent which should select one action (or arm) at every time step t until a finite horizon $t = 1, \dots, T$ and a sequence $r_{k,1}, \dots, r_{k,T}$ of unknown rewards associated with each arm k. Hence, at each time step t an agent, after choosing an action a_t, receives a reward $r_{a_t,t}$ which is drawn from an unknown probability distribution R_{a_t} (independently from the past). The effectiveness of the agent's strategy is evaluated by comparing it with the optimal action in expectation, with the aim of minimizing the pseudo-regret. The stochastic contextual bandit enriches the multi-armed bandit framework by incorporating context sets $\mathcal{S}$. The new objective of an agent is to identify the optimal policy within a class of policies that maps the context to the action.

3.2. Reinforcement learning

Multi-armed bandit and contextual bandit frameworks can be seen as particular cases of a full reinforcement learning problem. RL (Kaelbling et al. [30], Sutton and Barto [31]) is formulated as (completely observed) Markov Decision Process (MDP), where current actions can affect the next states. A MDP describe a decision-making process and it can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of states which contain information about the environment for each time step, $\mathcal{A}$ is a set of potential actions to be taken based on the states received by an agent and $\mathcal{R}$ is a set of rewards. T is a set of the transition probabilities $T_{sas'} = \mathbb{P}(S_t = s'|S_{t-1} = s, A_{t-1} = a)$ describing the dynamics of the environment, and $\gamma \in (0, 1)$ is a discount factor which is used to compute discounted return.

The most popular RL methods, such as Deep Q-learning, involve action-value functions $Q_\pi(s,a) = \mathbb{E}_\pi[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1}|S_t = s, A_t = a]$ to encode the policy π, which maps the states to the probabilities of selecting each action $a \in \mathcal{A}$. To solve the RL problem, the aim is to find a class of optimal policies that corresponds to finding the optimal action-value function $Q^*(s,a) = \max_\pi Q_\pi(s,a)$. Hence, an optimal policy can be reached by $\pi^*(a|s) \in \text{argmax}_a Q^*(s,a)$.

3.3. Exploration strategies

Exploration strategies are employed to balance the exploration–exploitation trade-off in the above settings. The most popular methods are, among others, ϵ-greedy, softmax, and Max-Boltzmann methods. In the case of Deep Q-learning (assuming full observable states), exploration strategies lead to exploring new actions rather than following a greedy policy, concerning the optimal action-value function, throughout the process. For instance, ϵ-greedy techniques balance the exploration–exploitation trade-off through a probability $\epsilon \in (0,1)$ such that

$$\pi_\epsilon(a|s) = \begin{cases} 1 - \epsilon_t + \frac{\epsilon_t}{|\mathcal{A}|}, & \text{if } a^* = \text{argmax}_{a \in \mathcal{A}} Q(s,a) \\ \frac{\epsilon_t}{|\mathcal{A}|}, & \text{otherwise} \end{cases}, \tag{1}$$

where $t = 1, \dots, T$ refers to time steps. Softmax exploration encompasses Boltzmann distribution function such that

$$\pi_B(a|s) = \frac{e^{\frac{Q(s,a)}{\kappa}}}{\sum_{l \in \mathcal{A}} e^{\frac{Q(s,l)}{\kappa}}}, \tag{2}$$

where κ regulates the trade-off between exploration and exploitation.

3.4. Recurrent reinforcement learning

Recurrent Reinforcement Learning is formulated as Partially Observable Markov Decision Process (POMDP), which deals with partially observable states instead of fully observable states as in MDP. POMDP (Hauskrecht [32]) can be seen as a generalization of MDP, allowing to model and reason about the uncertainty on the current state of the system in sequential decision problems (Ross et al. [33]). It is defined as a tuple $(\mathcal{S}, \mathcal{Z}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{Z}$ is a set of observations and $\mathcal{O}$ is the set of observation probabilities $O_{saz} = \mathbb{P}(Z_t = z|S_t = s, A_t = a)$. As in the previous case, Deep Q-learning encodes the policy by using Q-values which are estimated through a Deep Neural Network. Hence, the optimal policy can be reached by finding the optimal action-value function $Q^*(h,a) = \max_\pi Q_\pi(h,a)$, where $H_{t-1} = h$ is the previous information state.

4. Deep recurrent Q-learning for autonomous driving

In this section, we outline the proposed approach, based on a deep recurrent reinforcement learning framework, as a solution to autonomously control the steering angle of cars. Concurrently, we compare several exploration strategies to achieve an improved balance of exploration–exploitation trade-off, intending to enhance obstacle avoidance performance (Section 5). We also enrich our proposed method by incorporating task-specific rewards and implementing pre-processing techniques on the states. Fig. 1 shows the agent's network, the state representation, and the action set which are involved in the reinforcement learning framework. At each step, t, the state s_t is processed by the agent's network (in Fig. 1) which is used to estimate Q-values for each action. The agent chooses the action a_t which corresponds to the maximum of Q-values. Then, the reward value r_t is computed, and performing the action a_t, the environment provides the next state s_{t+1}. At a specific moment of the process, the agent's network is updated with traces of experience samples in a Bootstrapped Random Update fashion (Section 4.3).

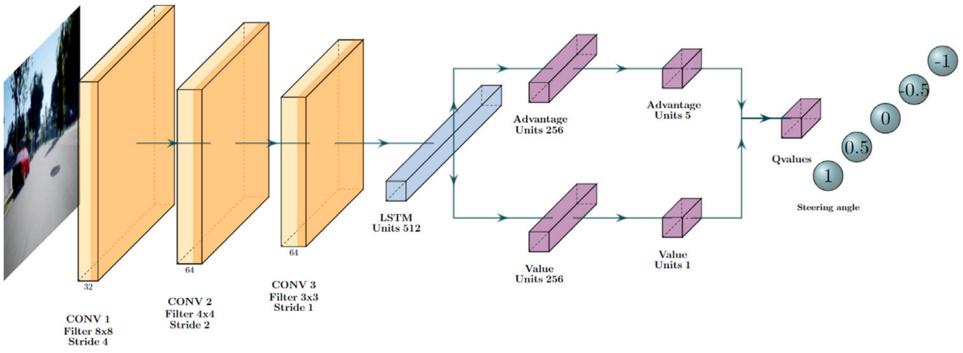

Fig. 1. Convolutional neural network with LSTM layer. Each pre-processed input image was processed by three convolutional layers and the resulting activations were performed through LSTM layer. Q-values were estimated by dividing the latter output into advantage and value fully-connected layers respectively and combining them together.

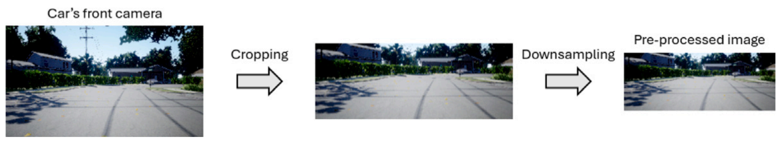

Fig. 2. Pre-processing of car's front camera images. The two-step pre-processing procedure is illustrated. Initially, each input image undergoes cropping at the top and bottom to concentrate the view of the road. Subsequently, downsampling is employed on the images before feeding them into the neural network.

4.1. Observation space

In our framework, the observations are represented by images captured by the car's front camera, extracting information from the surrounding environment. At each time step, an image is collected, providing a snapshot of the current road conditions ahead of the car. To highlight the road and the related obstacles, each image has been pre-processed by using cropping techniques. Additionally, to streamline image processing and reduce dimensionality, we apply downsampling on the cropped images, effectively eliminating non-essential information (see Fig. 2). The resulting size corresponds to 200 × 66 pixels.

4.2. Action space and reward

To avoid obstacles, we consider a set of discrete actions, each representing distinct values for the car's steering angle. The action set $\mathcal{A} = \{-1, -0.5, 0, 0.5, 1\}$ encompasses two levels of left steering, straight-ahead steering, and two levels of right steering. When an action is chosen, the car performs the related steering while keeping a constant speed.

We define a task-specific *Reward* by incorporating a measure of the distance between the position of the car (x_e) and the center of the roads (x_r), such that

$$r(s, s', a) = e^{-c \cdot \|x_e - x_r\|}, \tag{3}$$

where c is a positive constant and $r \in [0, 1]$ is the reward function. This formulation, proposed by Riboni et al. [24], and devised by Spryn et al. [34], is used to measure the goodness of steering angle actions, where the best action is achieved when the car is placed in the center of the road.

4.3. Agent's architecture

We consider the Double Dueling Deep Recurrent Q-Network architecture as the agent's network, where the weights are updated by using Bootstrapped Random Update. We propose to combine a modified loss function which leads to update weights only for the last observations of the episode trace (Lample and Chaplot [2]).

The agent's network combines convolutional and LSTM layers to estimate advantage $A(h_t, a; \vartheta, \alpha)$ and value function $V(h_t; \vartheta, \beta)$ for obtaining Q-value estimates (see Fig. 1) such that

$$Q(h_t, a; \theta) = V(h_t; \vartheta, \beta) + \Big(A(h_t, a; \vartheta, \alpha) - \frac{1}{|A|} \cdot \sum_{a'} A(h_t, a'; \vartheta, \alpha)\Big), \tag{4}$$

where $\theta = (\vartheta, \beta, \alpha)$ are the weights of the main network. The weights of the main network are updated by using a modified quadratic loss function, which is defined as

$$L(\theta_i) = \begin{cases} 0, & \text{if } i \le n_{err} \\ E_{(o_t, a_t, r_t, o_{t+1})}\big[(y_i - Q(h_t, a_t; \theta_i)^2)\big], & \text{otherwise} \end{cases}, \tag{5}$$

where $i = 1, \dots, T$ are the time steps of the LSTM layer. θ, θ' are the weights of the main network and the target network, respectively. n_{err} is the number of errors masked. The target y_i are estimated using a double estimator, as $y_i = r_t + \gamma Q(h_{t+1}, \text{argmax}_a Q(h_{t+1}, a; \theta_i); \theta'_i)$. During the weights updating process, Bootstrapped Random Updates are used as a sampling method. The choice of this method requires the LSTM's hidden state to be zeroed at the beginning of each update, and this peculiarity can make the learning of recurrent neural networks harder [14]. Consequently, to mitigate this challenge, only accurate gradients are propagated at each agent update through the network masking the first losses, to guarantee the update of the states with sufficient history [2]. The sample sequences are built by randomly drawing out episodes from the experience replay, followed by the definition of an experience trace. The length of this experience trace is regulated by a specific parameter, and the initial state is randomly selected from the sampled sequences (see Algorithm 1). Finally, the target network weights are updated by using the following rule

$$\theta' = \theta \cdot \eta + \theta' \cdot (1 - \eta)\ , \tag{6}$$

where η is a parameter that regulates the proportion of main network weights involved in the updated target network weights.

5. Exploration strategies for recurrent learning

The behavioral policies of our agents are contingent upon the selected exploration strategy. Together with the proposed method for autonomous driving (see Section 4.3), we conduct a comparative analysis of various exploration strategies, encompassing both deterministic

Algorithm 1 Bootstrapped Random Update with masking of errors

```
Buffer Size $n$, Trace Length $t$, Batch Size $b$, Loss Function $L$
Neural Network (Main) **MN**, Neural Network (Target) **TN**
Replay Memory $\mathcal{M} = \{\mathcal{S}_t, \mathcal{A}, \mathcal{S}_{t+1}, \mathcal{R}, \mathcal{T}\}$
**if** $length(\mathcal{M}) \geq \frac{n}{2}$ **then**    ▷ Update starts
    sample_episode = random_sample($\mathcal{S}_t$, n_sample = $b$)    ▷ Indicator of episodes
**end if**
**for** i in sample_episode **do**    ▷ Sampling
    sample_step = random_sample($\mathcal{S}_t$[i,], n_sample = 1)
    trace($\mathcal{S}_t$)= $\mathcal{S}_t$[i, sample_step : sample_step+t]    ▷ For each element of $\mathcal{M}$
**end for**
$main\_Q$ = **MN**(trace($\mathcal{S}_{t+1}$))    ▷ According to (4)
$target\_Q$ = **TN**(trace($\mathcal{S}_{t+1}$))    ▷ According to (4)
$double\_Q = target\_Q[, \text{argmax}_a\, main\_Q]$    ▷ $a \in \mathcal{A}$
$Y = trace(\mathcal{R}) + \gamma \cdot double\_Q \cdot (1 - trace(\mathcal{T}))$
$Q = \max_a \mathbf{MN}(trace(\mathcal{S}_t))$
$L_j = \frac{\sum_{i=1}^{t} w_i \cdot (Y_{j,i} - Q_{j,i})^2}{t}$    ▷ $w_i \in \{0,1\}$ with (5) condition; $\forall j \in \{1,\dots,b\}$
```

and adaptive approaches. Therefore, we present a theoretical overview of these strategies within deep recurrent reinforcement learning. We consider many ϵ-greedy and softmax methods, as well as their combinations. Within a partial observable process, the ϵ-greedy policy can be defined as

$$a_t = \begin{cases} \text{argmax}_a\, Q_t(h_t, a), & \text{with probability } 1-\epsilon \\ \text{any action(a)}, & \text{with probability } \epsilon \end{cases}. \tag{7}$$

The ϵ-greedy strategy balances exploration and exploitation by employing ϵ, which represents the exploration probability at each step. During the exploration phase, actions are selected randomly. Hence, how to adjust the ϵ probability poses one of the greatest challenges of RL, since it measures the degree to which the RL process should explore. The ϵ-greedy is widely used and can be adapted in various forms depending on how ϵ is determined, thus considering deterministic or stochastic approaches. However, a notable drawback of this strategy concerns the random policy during the exploration phase. Although exploration is intended to facilitate the discovery of new actions for optimal decision-making in the future, the use of a random policy introduces the risk of obtaining significantly worse actions. To address this issue, we compare ϵ-greedy strategies with Softmax and Max-Boltzmann Exploration methods.

Softmax is based on a different policy in which actions are drawn out from Boltzmann distribution. In the case of Deep Recurrent Q-Learning, we can define the Softmax policy as

$$\pi_B(a|h_t) = \frac{e^{\frac{Q(h_t,a)}{\kappa}}}{\sum_b e^{\frac{Q(h_t,b)}{\kappa}}}. \tag{8}$$

Despite this, the Softmax method entails continuous exploration throughout the process, wherein the degree of separability from a completely random policy is regulated by κ, known as the temperature, not leaving the possibility of integrating the greedy actions.

5.1. Deterministic ϵ-greedy

Using a deterministic strategy for ϵ-greedy, the evolution of the ϵ probability is fixed and exogenously defined independently of the ongoing process. We explore two deterministic approaches, including the simplest form known as constant ϵ-greedy. However, the latter may fall short of defining an optimal policy as it keeps the ϵ probability constant over time, resulting in the same amount of exploration throughout the process. Indeed, in the initial stages of learning, the agent should require more exploration than in the final stages due to the agent's greater uncertainty. To address this, we introduce an alternative deterministic approach where ϵ is modeled as a linear equation, the so-called Decreasing ϵ-greedy, allowing it to adapt based on the evolving dynamics of the process [24]. This method adjusts the decrease rate for ϵ, tailoring it differently for each number of steps. The system of the equations describing ϵ is given by

$$\epsilon_t = \begin{cases} 1, & \text{if } X_{steps} \leq n_{start} \\ \delta_0 + \delta_1 \cdot X_{steps}, & \text{if } X_{steps} > n_{start} + \epsilon_{ann} \\ \pi_0 + \pi_1 \cdot X_{steps}, & \text{otherwise} \end{cases}, \tag{9}$$

where X_{steps} represents the current step number, π_0, π_1 are the intercepts and δ_0, δ_1 are the slopes of the two linear functions. These coefficients are derived from the following equations

$$\pi_0 = \epsilon_{start} - \pi_1 \cdot n_{start}, \tag{10}$$

$$\pi_1 = -\frac{\epsilon_{start} - \epsilon_{last}}{\epsilon_{ann}}, \tag{11}$$

$$\delta_0 = \epsilon_{end} - \delta_1 \cdot n_{max}, \tag{12}$$

$$\delta_1 = -\frac{\epsilon_{last} - \epsilon_{end}}{n_{max} - \epsilon_{ann} - n_{start}}. \tag{13}$$

The ϵ probability is formulated using two different equations to get a different balance of the exploration–exploitation trade-off. The change point is built upon ϵ_{ann} parameter (expressed in frames), which specifies the number of steps after which ϵ should decrease more slowly. This method is indeed characterized by two linear trends: the initial phase is governed by a steeper decrease with higher values of ϵ, while the final phase adopts a more gradual trend with lower values of ϵ.

To gather sufficient experience memory, all exploration strategies are modified to ensure an initial phase of full exploration. The parameter n_{start} represents the upper limit in terms of steps that define the end of full exploration.

5.2. Value-difference based exploration

The VDBE strategy, introduced by Tokic [3] in Q-learning scenario, revolves around adjusting the ϵ probability based on Temporal Difference (TD) errors computed during the learning process. Unlike deterministic methods, VDBE is a data-driven ϵ-greedy approach that dynamically updates ϵ using information gained from the agent's learning. In this work, we extend this method to a Deep Q-learning scenario and specifically to the recurrent case. We consider a value of ϵ to be updated for each epoch as the difference between the previous ϵ value and a new piece of information obtained by the Boltzmann distribution function, as

$$f(h, a, v) = \frac{1 - e^{-\frac{\Delta_{err}}{v}}}{1 + e^{-\frac{\Delta_{err}}{v}}}, \text{ and} \tag{14}$$

$$\epsilon_{t+1} = \lambda \cdot f(h, a, v) + (1 - \lambda) \cdot \epsilon_t, \tag{15}$$

where λ and v respectively consist of the weights of the selected action and the inverse of sensitivity, which always expresses a positive constant. We define Δ_{err} as the difference between the optimal Q-values obtained after and before the update, such that

$$\Delta_{err} = Q_p(h_t, a^*) - Q_{p-1}(h_t, a^*), \tag{16}$$

where $p = (0, 1, \dots, n)$ are the n-epochs of the neural network and a^* is the optimal action chosen at the current step.

5.3. Bayesian model combination

Similar to VDBE, ϵ-Bayesian Model Combination (BMC) is a data-driven ϵ-greedy strategy that dynamically updates the ϵ value based on agent's data acquired during the RL process. The approach, proposed by Gimelfarb et al. [5], adopts a Bayesian perspective instead of relying on a heuristic approach, as seen in VDBE, for parameter tuning. ϵ-BMC leverages the strengths of Bayesian Model Combination, allowing a weighted combination of models. Specifically, such weights correspond to ϵ and $(1-\epsilon)$, as follows

$$\tilde{G} = (1-\epsilon)\cdot\tilde{G}^Q + \epsilon\cdot\tilde{G}^U \ , \tag{17}$$

where $\tilde{G}^Q$ e $\tilde{G}^U$ are, respectively, the two different models practiced in ϵ-greedy strategy: greedy and uniform, represented by

$$\tilde{G}^Q = r_{t+1} + \gamma \max_{a'\in A} Q_t(h_{t+1}, a') \ , \tag{18}$$

$$\tilde{G}^U = r_{t+1} + \gamma \frac{1}{|A|}\sum_{a'\in A} Q_t(h_{t+1}, a'). \tag{19}$$

Gimelfarb et al. [5] emphasized the distinction between the ϵ-BMC strategy and Bayesian Q-learning, wherein Q-values are modeled using Normal-Gamma priors. ϵ-BMC strategy concentrates solely on the variance distribution, employing Normal-Gamma prior. This method offers several advantages, including theoretical convergence guarantees within Q-learning. Furthermore, it proves to be a robust and efficient strategy for providing a good exploration balance.

Assuming a Normal distribution for return observations $Q_{h,a}$, expressed as

$$Q_{h,a}|m,\tau \sim N(\tilde{G}^m_t, \tau^{-1}) \ , \tag{20}$$

where m denotes the model chosen (greedy or uniform) and τ^{-1} represents the precision, alongside adopting a Normal-Gamma model for these parameters, results in the derivation of a Normal-Gamma posterior distribution. As mentioned earlier, this approach specifically centers on the prior distribution of return variance. By marginalizing the Normal-Gamma posterior, the resulting marginal distribution of τ, with parameters a_t and b_t, corresponds to

$$\tau|D \sim Gamma(a_t, b_t) \text{ , and} \tag{21}$$

$$a_t = a_0 + \frac{t}{2}, \quad b_t = b_0 + \frac{t}{2}\left(\tilde{\sigma}^2_t + \frac{\tau_0}{\tau_0 + t}(\hat{\mu}_t - \mu_0)^2\right). \tag{22}$$

The parameters $\hat{\mu}_t$ and $\tilde{\sigma}^2_t$ are calculated as the mean and the variance of the data of previously observed returns $D = \{Q_{h_i,a_i,i}, i = 1,\dots,t-1\}$ [5]. Consequently, in order to obtain the distribution of $Q_{h,a}|m,D$ by marginalizing over τ, it is possible to obtain t-distributed likelihood function as

$$P(Q_{h,a}|m,D) = \int_0^\infty P(Q_{h,a}|m,\tau)P(\tau|D)\,d\tau. \tag{23}$$

Solving Eq. (23), the kernel of a T-Student with three parameters $(\tilde{G}^m_t, \frac{a_t}{b_t}, 2a_t)$ is identified. Given the last result in Eq. (23), Gimelfarb et al. [5] computed the equation for obtaining the expected return $\mathbb{E}[Q_{h,a}|D]$ which involves the posterior distribution of $w|D$. This distribution characterizes the weights assigned to the greedy (18) and uniform model (19). The expected return is given by

$$\mathbb{E}[Q_{h,a}|D] = \int_0^1 \mathbb{E}[Q_{h,a}|w,D]\mathbb{P}(w|D)\,dw. \tag{24}$$

Eq. (24) combines the estimates of Q-values, derived from the Double Dueling Deep Recurrent Q-Network model and the uniform model, and the weights, which could be used to find a value for ϵ. Consequently, Eq. (24) can be reformulated as

$$\mathbb{E}[Q_{h,a}|D] = (1 - \mathbb{E}[w|D])\tilde{G}^Q_t + \mathbb{E}[w|D]\tilde{G}^U_t. \tag{25}$$

Subsequently, the study proceeds to determine the posterior distribution of the weights to establish a rule for updating ϵ. Since the Eq. (25), ϵ can be expressed as $\epsilon_t = \mathbb{E}[w|D]$. Based on this result, Gimelfarb et al. [5] used a technique (*i.e.* Dirichlet Moment-Matching) to approximate the posterior distribution for finding out the expected value of $w|D$, such that

$$\epsilon^{BMC}_t \approx E_{Beta(\alpha_t,\beta_t)}[w|D] = \frac{\alpha_t}{\alpha_t + \beta_t}. \tag{26}$$

Beta parameters (i.e. α_t and β_t) are obtained by the following system of equations

$$m_t = \frac{\alpha_t}{\alpha_t + \beta_t + 1}\,\frac{e^U_t(\alpha_t + 1) + e^Q_t\beta_t}{e^U_t\alpha_t + e^Q_t\beta_t} \ , \tag{27}$$

$$v_t = \frac{\alpha_t}{\alpha_t + \beta_t + 1}\,\frac{\alpha_t + 1}{\alpha_t + \beta_t + 2}\,\frac{e^U_t(\alpha_t + 2) + e^Q_t\beta_t}{e^U_t\alpha_t + e^Q_t\beta_t} \ , \tag{28}$$

$$r_t = \frac{m_t - v_t}{v_t - m^2_t} \ , \tag{29}$$

$$\alpha_{t+1} = m_t * r_t \ , \tag{30}$$

$$\beta_{t+1} = (1 - m_t) * r_t \ , \tag{31}$$

where e^U_t and e^Q_t are the evidence of a return under the distribution of $Q_{h,a}|m,D$ [5].

5.4. Max-Boltzmann Exploration

The fundamental idea behind Max-Boltzmann Exploration (MBE) is to leverage the Softmax policy for exploring actions, so as to balance exploration–exploitation trade-off with ϵ-greedy to take advantage of the strengths of both strategies. This approach was proposed by Wiering [35] and it integrates ϵ-greedy strategy with a difference in action sampling in the exploration phase, where actions are sampled over Boltzmann probability distribution as in Softmax policy. Indeed, at each time step, the action can be selected according to the following rule

$$a_t = \begin{cases} \text{argmax}_a\, Q_t(h_t, a), & \text{with probability } 1-\epsilon \\ \text{Softmax policy}, & \text{with probability } \epsilon \end{cases} . \tag{32}$$

The Max-Boltzmann policy effectively addresses a limitation of ϵ-greedy concerning the exploration probability distribution. The classical ϵ-greedy exploration assigns equal probabilities to all actions, thereby assigning too large probabilities to explore worse actions [35]. In contrast, including a Softmax policy into the ϵ-greedy method, can establish a probability distribution that deviates significantly from a uniform distribution, thereby avoiding a substantial decline in performance. The degree of separability from the uniform distribution is tied to the temperature parameter: with a high τ, the Softmax policy will converge to a random policy. On the other hand, Max-Boltzmann Exploration considers a probability of exploration by ϵ that overcomes the Softmax method, providing the flexibility to choose the extent of exploration.

An improvement of MBE could be drafted by VDBE-Softmax, wherein the ϵ probability is adjusted using the VDBE method, thus resulting in a heuristic strategy grounded in a data-driven approach [4]. The ϵ update is regulated like Eq. (15), totally integrated in MBE framework.

6. Experimental settings and simulation platform

In this section, we describe the experimental settings of our proposed method using AirSim as a simulation platform for autonomous driving. Specifically, our emphasis is on evaluating exploration strategies with both deterministic and adaptive approaches while controlling the performance in terms of collision avoidance. To conduct our analysis, at the beginning of the process, we position the car at specific points in the environment by randomly drawing out the coordinates from a predefined set of starting points. Hence, we assess the performance of RL agents in terms of learning, employing a set of ten training starting points. Additionally, we evaluate the predictive capabilities using another set of ten starting points (test set). The RL processes involve 1 million steps and each episode ends when a collision event occurs. The experiments were executed on a virtual machine with two GPUs Tesla M60 and Linux OS. AirSim simulator has been connected to Python. TensorFlow Compact V2, OpenAI, and OpenCV packages were used.

Fig. 3. AirSim NH environment. The neighborhood environment is shown through the AirSim simulation platform.

Table 1
Hyper-parameters of Double Dueling Deep Recurrent Q-Network.

Hyper-parameter	Range	Unit
Buffer size	$\{1000; 2000\}$	*Episode*
Batch size	10	*Episode*
Update rate	4	*Step*
Trace length	10	*Step*
Error masked	7	*Step*
State updated	3	*Step*
Starting update	999	*Episode*
End process	1,000,000	*Step*

6.1. AirSim simulator: Neighborhood environment

As previously mentioned, we use AirSim simulator to train and test Deep Double Dueling Recurrent Q-Learning models for learning self-driving cars. AirSim is an open-source program based on Unreal Engine, which offers various environments with several landscapes. With the aim of learning models for collision avoidance, we chose the AirSim NH environment. This environment represents a small, simplified, urban neighborhood with a rectangular shape. AirSim NH features a main street connected with side streets, including elements such as trees, parked cars, shadows, and lights, mimicking real-world scenarios (see Fig. 3). Shah et al. [36] provided an extensive analysis of AirSim highlighting its provision of realistic environments both in terms of physical and visual attributes. For executing Airsim, we employed OpenGL drivers and established a connection with Python. Within the environment, we set a single car, called PhysicXCar, along with specifying a single weather condition.

6.2. Models experimental setup

To facilitate a comprehensive comparison and evaluation of exploration strategies, we considered a predefined set of parameters based on Riboni et al. [24] results.

In Table 1 the parameter settings for the Double Dueling Deep Recurrent Q-Network algorithm[1] are shown. We varied the buffer size with two assigned values to assess the impact of experience buffer size changes on neural network optimization. Additionally, we opted for a soft update for the Target Network, where, at each updating step, the weights of the Target Network are updated with 0.1% of the main network weights. Furthermore, we initiated a completely random process for the first *Starting update* episodes.

6.2.1. Exploration strategies

We tested seven different exploration strategies, as detailed in Section 5, with a singular setting for each strategy. Table 2 shows all parameter settings used in the respective fourteen tests.

In particular, the parameters of BMC method align with those considered by Gimelfarb et al. [5] in their experimental settings. Similarly, the parameters of VDBE and VDBE-Softmax are based on experiments

[1] The code is available at https://github.com/ValentinaZangirolami/DRL.

Table 2
Hyper-parameters of Exploration strategies.

Strategy	Hyper-parameter	Value
Constant, MBE	ϵ	0.05
Decreasing ϵ-greedy	ϵ_{start}	1
Decreasing ϵ-greedy	ϵ_{last}	0.1
Decreasing ϵ-greedy	ϵ_{end}	0.01
VDBE, VDBE-Softmax	ν	1
VDBE, VDBE-Softmax	λ	0.2
BMC	α_0, β_0	25
BMC	a_0, b_0	250
BMC	μ_0	0
BMC	τ_0	1
Softmax, MBE, VDBE-Softmax	κ	0.1

carried out by Tokic [3] and Tokic and Palm [4]. While considering an exploration strategy involving an update regulated by the VDBE equation, the λ parameter was set equal to the inverse of the number of actions, and ν was assigned to a value within the range, avoiding extremes, as recommended by the authors. For Softmax, we assigned a moderate value for κ, following the results obtained by Tokic and Palm [4]. Consequently, we extended this value to MBE and VDBE-Softmax. Differently, the ϵ value assumed in constant ϵ-greedy is based on the values derived from the results of stochastic ϵ-greedy methods. The Decreasing ϵ-greedy method required a change in the updated structure of D3RQN model, since dependents on steps instead of episodes. Specifically, in Table 1 we set the starting point of the update to 999 episodes. In Decreasing ϵ-greedy, this parameter is adjusted in steps format (n_{start}) in Eqs. (10) and (13). We assigned a value of 50,000 steps to the n_{start} parameter and 400,000 steps for ϵ_{ann}. The remaining parameters of the Decreasing ϵ-greedy strategy are based on Riboni et al. [24].

Furthermore, we standardized the management of ϵ across all methods. According to the Decreasing ϵ-greedy strategy, we set ϵ equal to 1 to fill the experience buffer until the agent update starts.

6.3. Results and comparison

In this section, we present the results derived from the fourteen Double Dueling Deep Recurrent Q-Network models (as the settings outlined in Section 6) using the AirSim simulator, to evaluate the impact of the seven exploration strategies and the two buffer capacities on agent learning.

Fig. 4 illustrates the trends in terms of Reward values during the training process. All models exhibit an increasing trend, indicative of effective policy learning. However, deterministic strategies reach lower average rewards than other strategies. An intriguing observation is the notable contrast between ϵ-greedy strategies and those employing the Softmax method for exploration, such as Softmax, MBE, and VDBE-Softmax, which perform exceptionally well, reaching significantly higher values in the final steps. This effect might be attributed to the use of a different distribution function for exploration. As discussed in Section 5, the ϵ-greedy random policy may induce wrong actions, thereby leading to a substantial worsening of performance and, thus, the results obtained from the training may be misleading.

As illustrated in Fig. 5, stochastic strategies for ϵ-greedy exhibit very low values, even close to zero, whereas the deterministic method consists of much higher values throughout the learning process. Despite the VDBE-Softmax strategy being characterized by a Boltzmann distribution function with τ equal to 0.1, thus being very far from uniform distribution, the oscillation of ϵ is very similar to VDBE.

In Fig. 6, we analyze the spline of the loss differences for each model. It should be noted that, in Deep Reinforcement Learning tasks, the loss function is based on the difference between target values and prediction values. It can be generally observed that the gap between loss values before and after the update is too high with a larger size of

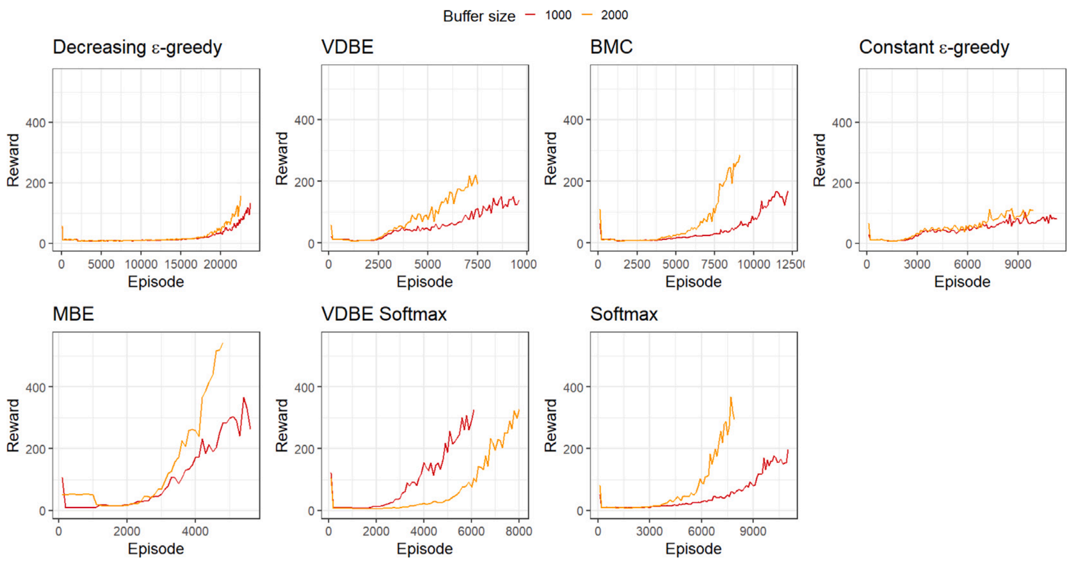

Fig. 4. Comparison of the exploration strategies with D3RQN agent. The training curves show the average reward per 100 episodes for each value of the buffer size. The horizontal axis and the vertical axis indicate, respectively, the number of episodes and the average reward.

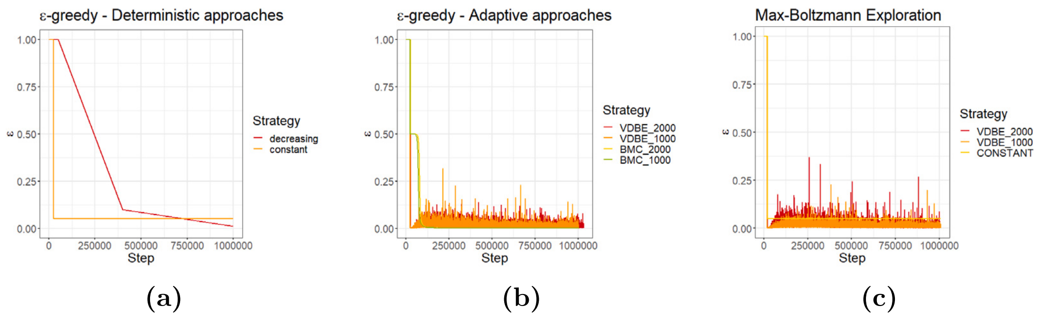

Fig. 5. Comparison of ϵ values for ϵ-greedy and Max-Boltzmann methods. The horizontal axis and the vertical axis indicate, respectively, the number of environment steps and the ϵ value. Figure **(a)** shows ϵ values per steps for deterministic ϵ-greedy strategies. Figure **(b)** shows ϵ values per step for adaptive ϵ-greedy strategies (BMC and VDBE) distinguished by buffer size. Figure **(c)** shows ϵ values per step for Max-Boltzmann Exploration methods (constant and VDBE) distinguished by buffer size.

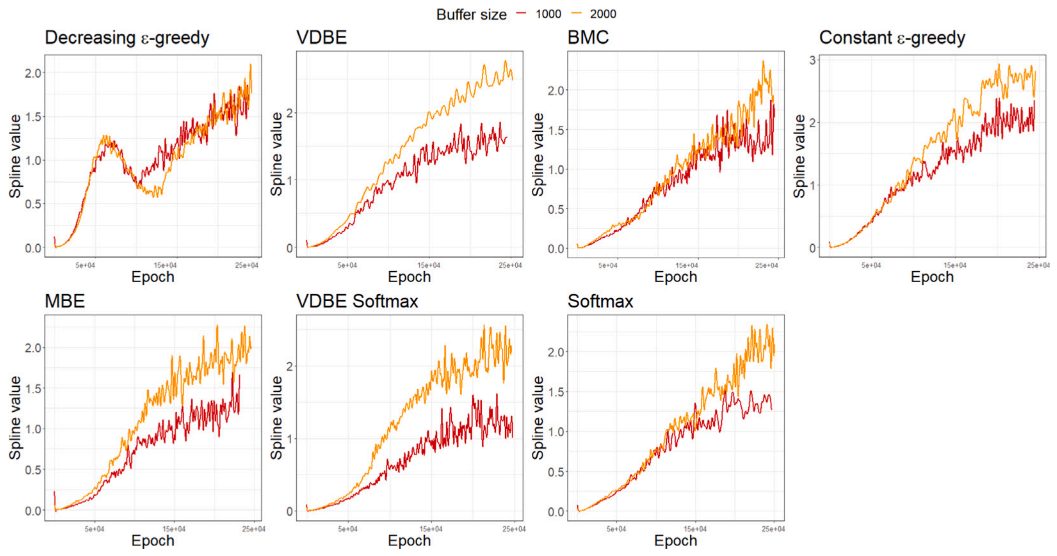

Fig. 6. Comparison of training loss differences for each exploration strategy and buffer size. The vertical axis denotes spline values of the absolute difference between the previous and the next loss. The horizontal axis indicates the training epochs.

Table 3
Training set evaluation. Performance of the agent over the training set with a buffer size equal to 2000. The agent was evaluated with 30 trials for each of the 10 training starting points. The agent performance is based on summary statistics of episode length and the collision-free rate (CFR).

Strategy	Average	Standard deviation	Min	CFR
Decreasing ϵ-greedy	1452.02	688.11	29	53.36%
Constant ϵ-greedy	1317.61	736.28	64	44.41%
VDBE	1795.24	474.16	**86**	80%
BMC	1769.24	532.5	34	80.47%
Softmax	1802.86	487.04	43	81.69%
VDBE-Softmax	**1917.26**	**325.36**	81	**91.92**%
MBE	1798.32	488.14	79	81.54%

Table 4
Training set evaluation. Performance of the agent over the training set with a buffer size equal to 1000. The agent was evaluated with 30 trials for each of the 10 training starting points. The agent performance is based on summary statistics of episode length and the collision-free rate (CFR).

Strategy	Average	Standard deviation	Min	CFR
Decreasing ϵ-greedy	1465.30	690.08	46	53.66%
Constant ϵ-greedy	1608.84	612.48	**114**	62%
VDBE	1728.66	565.27	71	76.09%
BMC	**1973.01**	**219.75**	7	**97.62**%
Softmax	1942.69	285.46	16	94.59%
VDBE-Softmax	1949.84	259.87	14	95.31%
MBE	1947.92	243.59	17	94.28%

Table 5
Test set evaluation. Performance of the agent over the test set with a buffer size equal to 2000. The agent was evaluated with 30 trials for each of the 10 test starting points. The agent performance is based on summary statistics of episode length and the collision-free rate (CFR).

Strategy	Average	Standard deviation	Min	CFR
Decreasing ϵ-greedy	1232.26	818.09	69	44.48%
Constant ϵ-greedy	1240.52	744.12	41	35.73%
VDBE	1593.84	700.01	40	68.64%
BMC	1604.47	718.93	40	72.47%
Softmax	**1666.6**	**655.72**	**92**	**75.92**%
VDBE-Softmax	1609.05	749.3	84	75.85%
MBE	1561.56	731.04	**92**	69.31%

buffer of experience. Furthermore, loss differences are too high when the agent improves performance in the long term.

In order to evaluate the goodness of steering angle prediction, we reported a summary of the performance gained from 300 episodes that were evaluated on 10 training starting points and on 10 test starting points respectively. We used different statistical indicators (among others, average, standard deviation, and minimum) of episode length and Collision-Free Rate (CFR) to evaluate the performance.

The results illustrated in Tables 3 and 4 show that all strategies, except VDBE, perform better with a smaller buffer size. In general, Collision-free rate and the average of steps have higher values together with a reduction in episode length variability, especially for the Softmax, VDBE-Softmax, MBE and ϵ-greedy BMC methods.

In Tables 5 and 6, the results obtained by 300 episodes for each model are exhibited. Since a single initial car position may not be sufficient to represent the performance and the generalization, ten different starting points are considered during testing.

We observed that performance remains very high on the test set, with a tiny discrepancy in comparison with the training set. The only drawback lies in the increasing step variability, which may suggest less stability. Generally, Softmax and related methods have better performance than ϵ-greedy strategies, except for the BMC method. Finally, in Fig. 7 we analyzed the optimality related to the action choice in order to understand which model was able to maximize the reward function at each step.

Table 6
Test set evaluation. Performance of the agent over the test set with a buffer size equal to 1000. The agent was evaluated with 30 trials for each of the 10 test starting points. The agent performance is based on summary statistics of episode length and the collision-free rate (CFR).

Strategy	Average	Standard deviation	Min	CFR
Decreasing ϵ-greedy	1316.54	789.73	59	48.95%
Constant ϵ-greedy	1508.11	722.97	29	59.79%
VDBE	1483.31	755.27	87	61.77%
BMC	**1821.72**	**509.57**	86	86.71%
Softmax	1724.11	641.66	48	82.25%
VDBE-Softmax	1791.01	573.97	**90**	**87.54**%
MBE	1738.52	636.68	87	84.14%

In general, the strategies that achieve the best results above involve more optimal actions. BMC, Softmax, MBE, and VDBE-Softmax strategies have a higher frequency of observations in the last bin, which contains the highest reward values. Furthermore, almost all of the methods have higher frequencies in the last bins when the buffer size equals 1,000. Particularly, Max-Boltzmann Exploration is the strategy that is most successful in terms of optimal actions, as almost 50% of the reward values are within the range [0.75,1].

7. Conclusions

This work investigates exploration strategies within a recurrent DRL framework addressing the issue of dealing with a lack of information about the state while finding a proper balance of the exploitation–exploration trade-off to relieve the uncertainty about the estimates. Partially observable systems have been less explored within the autonomous driving context, although they might be particularly suitable. In general, recurrent neural networks with Bootstrapped Random Updates often exhibit slow convergence bringing low learning efficiency within the DRL scenario. In an attempt to mitigate this issue, we combined the D3RQN model, featuring a Convolutional Recurrent Neural Network for estimating Q-values related to five steering angles, with a modified loss to speed up the learning during the process. Subsequently, several exploration strategies were evaluated to address the exploration–exploitation dilemma, employing both deterministic and adaptive approaches, among other ϵ-greedy and Softmax. While ϵ-greedy has been extensively studied in autonomous driving scenarios with its popular strategy of decreasing ϵ over time, Softmax and adaptive methods remain less explored. Indeed, we used a modified VDBE method for this framework as an adaptive strategy for ϵ-greedy and MBE, the so-called VDBE-Softmax, together with the ϵ-BMC method. Experimental results under deterministic approaches showed Softmax and MBE outperformed ϵ-greedy. However, adaptive strategies better balance exploration, enhancing collision avoidance. Notably, the uniform distribution used for sampling actions in ϵ-greedy exploration appeared to impair the agent learning in the training process, leading to worsening performance. Despite that, using Bayesian inference for ϵ-greedy updating yielded good results, even though the estimation of ϵ was close to zero after a few epochs and mitigating the potential negative impact of random sampling. In fact, the exploration drives the agent and hence the car to perform new steering angles which may be worse bringing the episode to an end because of the collision. This behavior can be especially emphasized by using a completely random policy during exploration, instead of using more sophisticated techniques, such as Softmax, which can be set to explore suboptimal actions avoiding worst actions, depending on its hyperparameter. These kinds of consequences are more close to our framework and in general to autonomous control systems, where the exploration can be helpful to find the best action introducing the risk of instability while the exploitation may achieve acceptable performance and maintain stability. In conclusion, the combination of Boltzmann distribution and ϵ-greedy proved effective in leading exploration and facilitating learning in the environment, outperforming other methods and achieving more optimal actions.

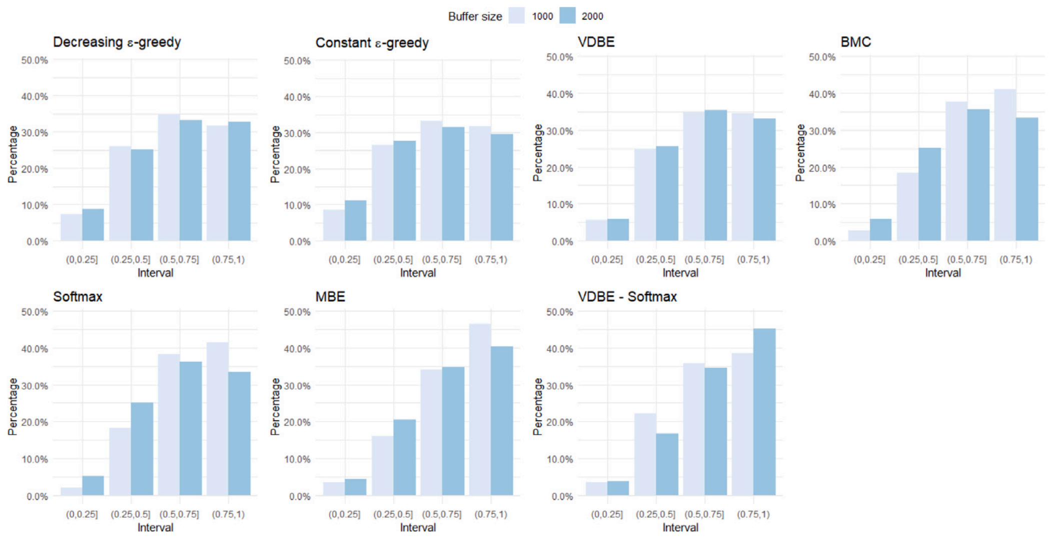

Fig. 7. Comparison of test reward values for each exploration strategy and buffer size. Each barplot shows the percentage of reward values falling within each range. The horizontal axis indicates different intervals of reward values in ascending order. The last bin contains the maximum values of reward meaning the optimal choices.

7.1. Current limitation and potential future enhancements

Our proposed autonomous driving framework may face challenges when subjected to testing in real-world environments. Although we studied the impact of balancing exploration and exploitation with different techniques on choosing the optimal steering angles to avoid collisions, our approach was constrained by considering only a small discrete set of actions. Real environments should demand a higher granularity of steering angles for effective and secure navigation. Additionally, we control the speed by setting deterministic thresholds. However, our model can still be applicable in real scenarios by using the estimated weights of our neural network. Alternatively, the network can serve as a pre-trained model for transfer learning or feature extraction techniques.

Future research could continue to explore Max-Boltzmann exploration strategies. As proposed by Gimelfarb et al. [5], it would be interesting to incorporate Bayesian inference for estimating ϵ probability, as with ϵ-greedy BMC method.

CRediT authorship contribution statement

Valentina Zangirolami: Writing – original draft, Validation, Methodology, Conceptualization. **Matteo Borrotti:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the link to our data/code at the following link: https://github.com/ValentinaZangirolami/DRL.

References

[1] P. Auer, Using confidence bounds for exploitation-exploration trade-offs, J. Mach. Learn. Res. 3 (2002) 397–422.
[2] G. Lample, D.S. Chaplot, Playing FPS games with deep reinforcement learning, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017, pp. 2140–2146.
[3] M. Tokic, Adaptive ϵ-Greedy exploration in reinforcement learning based on value differences, in: KI 2010: Advances in Artificial Intelligence, Springer Berlin Heidelberg, 2010, pp. 203–210.
[4] M. Tokic, G. Palm, Value-difference based exploration: Adaptive control between epsilon-Greedy and Softmax, in: KI 2011: Advances in Artificial Intelligence, Springer Berlin Heidelberg, 2011, pp. 335–346.
[5] M. Gimelfarb, S. Sanner, C.-G. Lee, Epsilon-BMC: A Bayesian ensemble approach to Epsilon-Greedy exploration in model-free reinforcement learning, in: Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, in: Proceedings of Machine Learning Research, vol. 115, 2020, pp. 476–485.
[6] A. Ortiz, H. Al-Shatri, X. Li, T. Weber, A. Klein, Reinforcement learning for energy harvesting point-to-point communications, in: 2016 IEEE International Conference on Communications, ICC, 2016, pp. 1–6.
[7] F. Cruz, P. Wüppen, A. Fazrie, C. Weber, S. Wermter, Action selection methods in a robotic reinforcement learning scenario, in: 2018 IEEE Latin American Conference on Computational Intelligence (la-CCI), 2018, pp. 1–6.
[8] K. Minwoo, K. Jongyun, J. Minjae, O. Hyondong, Towards monocular vision-based autonomous flight through deep reinforcement learning, Expert Syst. Appl. 198 (2022).
[9] C. Núñez Molina, J. Fernández-Olivares, R. Pérez, Learning to select goals in automated planning with Deep-Q learning, Expert Syst. Appl. 202 (2022).
[10] H. Alavizadeh, H. Alavizadeh, J. Jang-Jaccard, Deep Q-learning based reinforcement learning approach for network intrusion detection, Computers 11 (3) (2022).
[11] Y. Zhang, P. Sun, Y. Yin, L. Lin, X. Wang, Human-like autonomous vehicle speed control by deep reinforcement learning with Double Q-Learning, in: 2018 IEEE Intelligent Vehicles Symposium, IV, 2018, pp. 1251–1256.
[12] K. Min, H. Kim, K. Huh, Deep Q learning based high level driving policy determination, in: 2018 IEEE Intelligent Vehicles Symposium, IV, 2018, pp. 226–231.
[13] H. Xuefeng, H. Hongwen, W. Jingda, P. Jiankun, L. Yuecheng, Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle, Appl. Energy 254 (2019) 113708.
[14] M. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, in: 2015 Aaai Fall Symposium Series, (6) 2015.
[15] C. Romac, V. Béraud, Deep recurrent Q-learning vs deep Q-learning on a simple partially observable Markov decision process with minecraft, 2019.
[16] J. Zeng, J. Hu, Y. Zhang, Adaptive traffic signal control with deep recurrent Q-learning, in: 2018 IEEE Intelligent Vehicles Symposium, IV, 2018, pp. 1215–1220.
[17] J. Ou, X. Guo, M. Zhu, W. Lou, Autonomous quadrotor obstacle avoidance based on dueling double deep recurrent Q-learning with monocular vision, Neurocomputing 441 (2021) 300–310.
[18] Y. Xu, J. Yu, R. Buehrer, Dealing with partial observations in dynamic spectrum access: Deep recurrent Q-networks, in: MILCOM 2018 - 2018 IEEE Military Communications Conference, MILCOM, 2018, pp. 865–870.
[19] Y. Wu, S. Liao, X. Liu, Z. Li, R. Lu, Deep reinforcement learning on autonomous driving policy with auxiliary critic network, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–11.
[20] A. Santara, S. Rudra, S.A. Buridi, M. Kaushik, A. Naik, B. Kaul, B. Ravindran, MADRaS: Multi agent driving simulator, J. Artificial Intelligence Res. 70 (2021) 1517–1555.

[21] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, A.M. López, Multimodal end-to-end autonomous driving, IEEE Trans. Intell. Transp. Syst. 23 (1) (2022) 537–547.
[22] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, M. Kwiatkowska, Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 7344–7350.
[23] C. Pilz, G. Steinbauer, M. Schratter, D. Watzenig, Development of a scenario simulation platform to support autonomous driving verification, in: 2019 IEEE International Conference on Connected Vehicles and Expo, ICCVE, 2019, pp. 1–7.
[24] A. Riboni, A. Candelieri, M. Borrotti, Deep autonomous agents comparison for Self-Driving Cars, in: Proceedings of the 7th International Conference on Machine Learning, Optimization and Big Data - LOD, 2021, pp. 201–213.
[25] N. Deshpande, D. Vaufreydaz, A. Spalanzani, Behavioral decision-making for urban autonomous driving in the presence of pedestrians using Deep Recurrent Q-Network, in: 2020 16th International Conference on Control, Automation, Robotics and Vision, ICARCV, 2020, pp. 428–433.
[26] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, D. Cao, Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning, IEEE Access 8 (2020) 177804–177814.
[27] L. Rodman, On the many-armed Bandit problem, Ann. Probab. 6 (3) (1978) 491–498.
[28] P. Whittle, Multi-armed bandits and the gittins index, J. R. Stat. Soc. Ser. B Stat. Methodol. 42 (2) (1980) 143–149.
[29] S. Bubeck, N. Cesa-Bianchi, Regret analysis of stochastic and nonstochastic multi-armed bandit problems, Found. Trends® Mach. Learn. 5 (1) (2012) 1–122.
[30] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, J. Artificial Intelligence Res. 4 (1) (1996) 237–285.
[31] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 2014.
[32] M. Hauskrecht, Value-function approximations for partially observable Markov decision processes, J. Artificial Intelligence Res. 13 (2000) 33–94.
[33] S. Ross, J. Pineau, B. Chaib-draa, P. Kreitmann, A Bayesian approach for learning and planning in partially observable Markov decision processes, J. Mach. Learn. Res. 12 (48) (2011) 1729–1770.
[34] M. Spryn, A. Sharma, D. Parkar, M. Shrimal, Distributed deep reinforcement learning on the cloud for autonomous driving, in: IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), 2018, pp. 16–22.
[35] M. Wiering, Explorations in Efficient Reinforcement Learning (Ph.D. thesis), University of Amsterdam, 1999.
[36] S. Shah, A. Kapoor, D. Dey, C. Lovett, AirSim: High-fidelity visual and physical simulation for autonomous vehicles, Field Serv. Robot. (2017) 621–635.