Impact of multi-armed bandit strategies on deep recurrent reinforcement learning

Valentina Zangirolami[1] and Matteo Borrotti[1]

[1]University of Milano-Bicocca, Milan, Italy

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. Such as when only 2D images are considered as input in a RL approach used for finding the optimal action 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 scenario. More precisely, the final aim is to investigate the effects of using both stochastic and deterministic multi-armed bandit strategies coupled with a Deep Recurrent Q-Network. Additionally, we adapted and evaluated the impact of an innovative method to improve the learning phase of the underlying Convolutional Recurrent Neural Network. We aim to show that adaptive stochastic methods for exploration better approximate the trade-off between exploration and exploitation as, in general, Softmax and Max-Boltzmann strategies are able to outperform ϵ-greedy techniques.

Keywords— Exploration strategies, Deep Reinforcement Learning, Autonomous driving

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 in learning via interactions with an unknown environment, and exploitation, which consists in optimizing the objective function given accumulated information. The latter can be studied using the (stochastic) control theory, while the former relies on the theory of statistical learning.

In such a scenario, a 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 Auer (2002).

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 exploration-exploitation trade-off. For this purpose, our contribution is threefold. First, deep analysis and comparison of different exploration strategies on partially observable systems is conducted. In particular, we compared several ϵ-greedy and softmax approaches, including a Bayesian perspective and an estimation of the agent's uncertainty to adapt the ϵ probability. Originally, these methods were defined by Tokic (2010), Tokic and Palm (2011) and Gimelfarb et al. (2020) over a full observability system states. All strategies were adapted to the recurrent framework. Second, we adapted the sampling strategy for updating neural network weights proposed by Lample and Chaplot (2017) for the Bootstrapped Random Update sampling technique. Lample and Chaplot (2017) developed the 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. Third, the simulation study is focused on the prediction of steering wheel angle in autonomous driving systems. In this application, we considered stochastic exploration strategies on partially observable systems for balancing exploration, which is a central issue in dealing with 3D environments.

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 an overview of deep recurrent reinforcement learning and a description of the used model. Section 4 describes exploration strategies with an overview of the Multi-Armed Bandit issues in RL. Section 5 specifies the experimental settings for the use of AirSim simulator, the parameters of exploration strategies and DRL model. Section 6 shows the results obtained from previous experiments on training and test set and a deep analysis of the different effects of exploration strategies. Section 7 concludes this paper with final considerations and outlines 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

The aim of this paper falls on the study of the agent's uncertainty about the environment as "How to explore" is one of the radical issue of RL. Specifically, we considered deterministic and stochastic adaptive approaches for ϵ-greedy and Softmax techniques to face exploration-exploitation trade-off. 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. (2016) 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. (2020) and Tokic (2010) proposed two novel adaptive methods for ϵ-greedy on full-observable domain and used training information to update ϵ probability. Tokic (2010) 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. (2020) used Bayesian Inference to estimate the ϵ value with theoretical convergence guarantee and they proceeded to compare this strategy with deterministic and adaptive ϵ-greedy approaches. However, both papers include a limited number of strategies since Gimelfarb et al. (2020) only make comparisons between ϵ-greedy strategies while Tokic (2010) only includes constant ϵ-greedy while Softmax. Tokic and Palm (2011) proposed an extension of Max-

Boltzmann Exploration in a full-observable states domain which combines Softmax and value-based adaptive ϵ-greedy. As with the previous cases, Tokic and Palm (2011) compared this novel strategy with constant ϵ-greedy, VDBE and Softmax regardless of decreasing ϵ-greedy and other adaptive strategies. Cruz et al. (2018) 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 (2011) and Tokic (2010). 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 analysed 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 matter has recently enticed researchers to apply Deep Q-Learning (DQL) methods on video-games and autonomous driving tasks, which all require Convolutional Neural Network 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. (2022) 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. (2022) showed that DDQN and D3QN overcome the performance of the baseline DQN structure, in which D3QN learns better policies than other methods. Some works Núñez-Molina et al. (2022), Alavizadeh et al. (2022), Zhang et al. (2018), Min et al. (2018), Xuefeng et al. (2019) evaluated Deep Q-Learning in several RL applications. However, DQN and its extensions assume the full observability of the states which falls in real world and, in general, 3D environment. The real-world environment is full of 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 (2015) 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 (2015) 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 (2015) showed how DRQN updates might trigger some problems with the learning of functions which could be reflected in the final performance. Other papers Romac and Béraud (2019), Zeng et al. (2018), Ou et al.

(2021) and Xu et al. (2018) 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 (2017) proposed a novel method for Deep Recurrent Q-Network so as to overcome challenges during agent's update. Consistently, Lample and Chaplot (2017) proposed a technique based on error mask in the optimization phase in order to update neural network weights with enough history of observations. Lample and Chaplot (2017) tested their techniques coupled with Bootstrapped Sequential Update as sampling strategy. In this work, we extend the contribution of Lample and Chaplot (2017) 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 which generate iterative image and perform actions in real-time. Wu et al. (2021) and Santara et al. (2021) 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 a number of real-world peculiarities like parked cars, static obstacles, animals and vegetations. Xiao et al. (2022) and Michelmore et al. (2020) 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. (2019) 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 literature, there are some works of DRL for autonomous driving. Riboni et al. (2021) 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. (2020) proposed a DRQN model with descending ϵ-greedy to control car's steering angle and speed, as Liao et al. (2020) 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 Deep recurrent reinforcement learning

Deep Recurrent Reinforcement Learning is a machine learning approach that combines Reinforcement Learning and recurrent structures of Deep Learning. In general, Reinforcement Learning environments are formulated as Markov Decision Process (MDP), which could be solved by estimating Q-values using neural networks. MDP assumes the full observability of states whose validity could be compromised in real world. The use of recurrent structures manages to overcome these problems through a system based on hidden states. Such system guarantees the ability to manage situations in which full knowledge of a state is not available due to real world conditions, thus possibly bringing to a limitation of visibility or to the introduction of noise (Wiering and Van Otterlo, 2012).

MDP describe a decision-making process based on Markov chain properties and it can be defined as a tuple (S, A, T, R, γ), where S (set of states) and A (set of actions) are the main quantities allowing the interaction between agent and environment. The other elements represent the T transition model, the R Reward function and the discount factor $\gamma \in (0, 1)$.

At each time step, the agent takes the action $a \in A$ based on the state $s \in S$ received from the environment and ends up in the next state $s' \in S$ produced by T. Finally, the agent receives $r \in R$ from the environment given the current action.

The main objective lies in maximizing the sum of the future rewards $\mathbb{E}[\sum_t^\infty \gamma^t R(s_t, a_t)]$ to find out the optimal policy $\pi(a|s)$, which defines the action distribution with a given state. A way to reach the optimal action through π consists in estimating the optimal action-value function $Q^*(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma \max_a Q^*(s_{t+1}, a_{t+1})|s_t, a_t]$ that is computed according to the Bellman optimality equation.

3.1 Deep Recurrent Q-Network

Deep Recurrent Q-Learning could be used to solve Partially Observable Markov Decision Process (POMPD). It best captures the dynamics of many real worlds environment (Hausknecht and Stone, 2015) in which the agent receives partial state information by the observation $o \sim O(s)$, where $s \in S$. POMPD involves an underlying hidden state-based system which uses belief state as an estimation of the current state for choosing the best action.

Deep Q-network may not be involved in solving POMPD framework while Deep recurrent Q-network could approximate the underlying hidden state-based system. Hausknecht and Stone (2015) introduced the Deep recurrent Q-network by using Deep Q-network and recurrent layer, in which the latter is attached at the end of the convolutional layers. Instead, Lample and Chaplot (2017) proposed a method to mask the gradient of neural network for improving the learning of neural network functions.

The Deep recurrent Q-network model estimates $Q(o_t, a_t, h_{t-1})$, now depending on current observation (o_t), previous hidden state (h_{t-1}) and current action (a_t). However, we considered Double Dueling Deep Recurrent Q-Network (D3RQN), where the recurrent layers are represented by Long Short Term Memory (LSTM) layer. Thus, our model could be used to estimate $Q(h_t, a_t)$ instead of $Q(s_t, a_t)$, where $h_t = LSTM(o_t, h_{t-1})$.

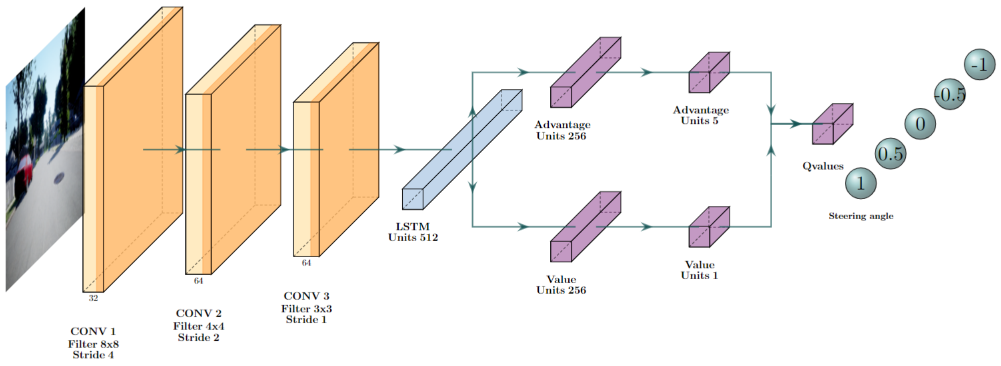


Figure 1: Convolutional Neural Network with LSTM layer. Each input image was processed by three convolutional layers and the resulting activations were performed through LSTM layer. Q-values were estimated by dividing the latter output into advantage and value fully-connected layers respectively and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

In figure 1, the D3RQN model is shown. As it can be observed, the output of the LSTM layer is divided by Advantage and Value layers that generate Q-values as expressed in Eq. (1).

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

During the weights updating, Bootstrapped Random Updates are used as sampling method. The choice of this method requires the LSTM's hidden state to be zeroed at the beginning of each update. However, this peculiarity can make the learning of recurrent neural network harder (Hausknecht and Stone, 2015). For this reason, only accurate gradients are propagated at each agent update through the network masking the first losses, in order to guarantee the update of the states with sufficient history (Lample and Chaplot, 2017).

The weights of Convolutional recurrent neural network are updated by Adam optimizer with a learning rate equal to 0.0001 in which loss function is expressed as Eq. (2).

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

where $i = 1, \ldots, T$ are the time steps and T is the length of the time sequence, θ, θ' are, respectively, weights of the prediction network and the target network and n_{err} is the number of errors masked. The target y_i is defined by $y_i = r_t + \gamma Q(h_{t+1}, \operatorname{argmax}_a Q(h_{t+1}, a; \theta_i); \theta'_i)$.

The target network weights are updated for each updating step according to the following rule:

$$\theta' = \theta \cdot \eta + \theta' \cdot (1 - \eta) \tag{3}$$

where η is a parameter can be interpreted as *"weight"* of a weighted average between the weights of the main network and the target network.

4 Multi-Armed Bandit strategies

A crucial aspect of Reinforcement Learning is how to find a balance of exploitaton-exploration trade off. This matter is related to the Multi-Armed Bandit problem and it is crucial to finding optimality in accordance to achieving maximum reward.

There are many methods in the literature to approximate the exploration-exploitation trade-off, amongst which the most cited are ϵ-greedy and softmax techniques (Sutton and Barto, 2014).

The ϵ-greedy strategy balances exploration and exploitation by using ϵ, which represents the exploration probability on each step. The exploration phase is regulated by extracting action randomly. The ϵ-greedy policy may be defined as Eq. (4).

$$a_t = \begin{cases} \operatorname{argmax}_a Q_t(h_t, a), & \text{with probability } 1 - \epsilon \\ \text{any action(a)}, & \text{with probability } \epsilon \end{cases} \tag{4}$$

How to adjust the ϵ probability is one of the greatest challenges of RL, since it is the quantity that describes how much the process should explore or exploit.

The ϵ-greedy is the most used strategy that can be applied in different forms related to how ϵ is determined, thus considering deterministic or stochastic approaches. However, there is a disadvantage that concerns the random policy in the exploration phase; *"exploration"*, indeed, is supposed to ensure the possibility to explore new actions with the aim of choosing optimal actions in the future. Through a random policy, the agent explores a random action which may not be optimal but which introduces the possibility to learn useful actions for the future in order to achieve long-term optimality.

For this reason, we considered a comparison between several strategies, also analyzing the Softmax technique and Max-Boltzmann Exploration.

Softmax is based on a different policy in which actions are extracted by Boltzmann distribution, as reported in Eq. (5).

$$\pi(a|h_t) = \frac{e^{\frac{Q(h_t,a)}{\kappa}}}{\sum_b e^{\frac{Q(h_t,b)}{\kappa}}} \tag{5}$$

Despite that, Softmax method involves continuous exploration throughout the process, where the degree of separability of a completely random policy is controlled by κ, known as the temperature, not leaving the possibility of integrating the greedy actions.

4.1 Deterministic ϵ-greedy approach

Using a deterministic strategy for ϵ-greedy, the evolution of the ϵ probability is handled exogenously. We have considered two deterministic approaches, among other the easiest form whose ϵ is constant over time. Sometimes, this method fails to define an optimum policy because it may be more efficient to adjust ϵ differently as the process evolves.

In general, in the initial stages of learning, it is assumed that more exploration is required than in the final stages due to the agent's greater uncertainty about the environment. For this reason, this paper includes another deterministic approach, in which ϵ is expressed as a linear equation in order to adapt ϵ on the basis of the evolution of the process (Riboni et al., 2021). This method adjusts the decrease of ϵ differently for each number of steps, where the system of the equations describing the ϵ linear trend is given by Eq. (6).

$$\epsilon_t = \begin{cases} 1, & \text{if } X_{steps} \leq n_{start} \\ \delta_0 + \delta_1 \cdot X_{steps}, & \text{if } X_{steps} > n_{start} + \epsilon_{ann} \\ \pi_0 + \pi_1 \cdot X_{steps}, & \text{otherwise} \end{cases} \tag{6}$$

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 obtained from the equations (7), (8), (9), (10).

$$\pi_0 = \epsilon_{start} - \pi_1 \cdot n_{start} \tag{7}$$

$$\pi_1 = -\frac{\epsilon_{start} - \epsilon_{last}}{\epsilon_{ann}} \tag{8}$$

$$\delta_0 = \epsilon_{end} - \delta_1 \cdot n_{max} \tag{9}$$

$$\delta_1 = -\frac{\epsilon_{last} - \epsilon_{end}}{n_{max} - \epsilon_{ann} - n_{start}} \tag{10}$$

The ϵ probability is implemented as two different equations in order to balance differently exploration-exploitation trade-off. The change point is built upon ϵ_{ann} parameter (expressed in frames), which defines the number of steps after which ϵ should decrease more slowly. This method is indeed characterized by two linear trends: the first phase is regulated by a steeper decrease with higher values of ϵ and the final phase involves a flattened trend with lower values of ϵ.

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

4.2 Value-Difference Based Exploration

The Value Difference Based Exploration (VDBE) strategy concerns with the fitting of ϵ probability on the basis of Temporal Difference Errors (TD), which are calculated during the learning process Tokic (2010). Unlike deterministic methods, VDBE is a data-driven ϵ-greedy approach that updates ϵ with the information gained from the agent's learning at each environmental step.

The ϵ value is updated for each epoch by the previous value and the new piece of information obtained by the Boltzmann distribution function, as expressed in Eq. (12).

$$f(h, a, \nu) = \frac{1 - e^{-\frac{\Delta_{err}}{\nu}}}{1 + e^{-\frac{\Delta_{err}}{\nu}}} \tag{11}$$

$$\epsilon_{t+1} = \lambda \cdot f(h, a, \nu) + (1 - \lambda) \cdot \epsilon_t \tag{12}$$

where λ and ν respectively consist of the weights of the selected action and the inverse of sensitivity, which always expresses a positive constant.

In order to obtain a Δ_{err}, the difference between the optimal Q-values obtained after and before the update is defined for each optimization step (see Eq. (13)).

$$\Delta_{err} = Q_t(h_t, a^*) - Q_{t-1}(h_t, a^*) \tag{13}$$

where $t = (0, 1, \dots, n)$ are the n-epochs of the neural network and a^* is the optimal action chosen at the current step. The updating of neural network weights may be handled to estimate $Q_t(h_t, a^*)$, which takes place as described in the previous section.

4.3 Bayesian Model Combination

Like VDBE, Bayesian Model Combination (BMC) is a data-driven ϵ-greedy strategy which updates the ϵ value with the training information. Differently, this strategy is based on the bayesian inference in which training information is extrapolated through Q-values.

In accordance with the ϵ-greedy policy, the combination model can be specified by Eq. (14).

$$\tilde{G} = (1 - \epsilon) \cdot \tilde{G}^Q + \epsilon \cdot \tilde{G}^U \tag{14}$$

Where $\tilde{G}^Q$ e $\tilde{G}^U$ are, respectively, the two different models practiced in ϵ-greedy strategy: greedy and uniform, as represented in Eq. (15) and Eq. (16).

$$\tilde{G}^Q = r_{t+1} + \gamma \max_{a' \in A} Q_t(h_{t+1}, a') \tag{15}$$

$$\tilde{G}^U = r_{t+1} + \gamma \frac{1}{|A|} \sum_{a' \in A} Q_t(h_{t+1}, a') \tag{16}$$

Gimelfarb et al. (2020) proposed ϵ-BMC strategy for Q-learning scenarios with a theoretical convergence guarantee, where the objective is to find an appropriate estimate of ϵ through bayesian perspective.

The main assumption refers to the normal distribution of returns as expressed in Eq. (17), over the mean $\mu_{h,a}$ and precision $\tau_{h,a}$, i.e. the inverse of its variance Dearden et al. (1998).

$$R_{h,a} \sim N(\mu_{h,a}, \tau_{h,a}) \tag{17}$$

The second assumption refers to the prior distribution of μ and τ, as is in the case of Dearden et al. (1998), which may be expressed by the Eq (18).

$$\mu, \tau \sim NormalGamma(\mu_0, \tau_0, a_0, b_0) \tag{18}$$

$$Q_{h,a}|\mu, \tau \sim N(\mu, \tau^{-1}) \tag{19}$$

where $(\mu_0, \tau_0, a_0, b_0)$ is a tuple of hyperparameters and $Q_{h,a}|\mu, \tau$ are independent and identically distributed. In order to compute the posterior distribution, the second assumption even involves that the prior distributions of $(\mu_{h,a}, \tau_{h,a})$ and $(\mu_{h',a}, \tau_{h',a})$ are independent for $h \neq h'$ and $a \neq a'$. The condition of independence related to the Eq. (19), as in the standard Q-learning, may be also violated in our framework.

In order to compute the likelihood for obtaining the posterior distribution, the last assumption refers to the distribution of Q-values conditional on the model m and τ as expressed in Eq. (20).

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

The mean $\tilde{G}_t^m$ would be obtained by equations Eq. (15) and Eq. (16), where $m = \{Q, U\}$ represents the models.

Given the prior distribution in Eq. (18) and after computing the likelihood with the data D, the marginal posterior distribution is a Gamma with parameters a_t and b_t, as expressed in Eq. (21).

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

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

The parameters $\hat{\mu}_t$ and $\tilde{\sigma}_t^2$ are calculated as the mean and the variance of the data of previously observed returns D Gimelfarb et al. (2020).

Then, 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 expressed in Eq. (23).

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

By solving Eq. (23), the kernel of a T-Student with three parameters $(\tilde{G}_t^m, \frac{a_t}{b_t}, 2a_t)$ is identified. Since the last result in Eq. (23), Gimelfarb et al. (2020) show how to obtain the expected return $\mathbb{E}[Q_{h,a}|D]$ involving the posterior distribution of $w|D$, which represents the posterior distribution of the weights assigned to the greedy (15) and uniform model (16).

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

Eq. (24) combines the estimates of Q-values, could be obtained by Double Dueling Deep Recurrent Q-Network model and the uniform model, with the weights, which could be used to find a value for ϵ.

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

After that, the study proceeds to identify the posterior distribution of the weights in order to find a rule to update ϵ. Since the Eq. (25), ϵ could be expressed as $\epsilon_t = \mathbb{E}[w|D]$.

Given this last result, Gimelfarb et al. (2020) proposed to use a technique to approximate the posterior distribution for just finding out the expected value of $w|D$.

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

Beta parameters (i.e. α_t and β_t) are calculated using the Dirichlet Moment-Matching technique. We reported the equations system as follows.

$$m_t = \frac{\alpha_t}{\alpha_t + \beta_t + 1} \frac{e_t^U(\alpha_t + 1) + e_t^Q\beta_t}{e_t^U\alpha_t + e_t^Q\beta_t} \tag{27}$$

$$v_t = \frac{\alpha_t}{\alpha_t + \beta_t + 1} \frac{\alpha_t + 1}{\alpha_t + \beta_t + 2} \frac{e_t^U(\alpha_t + 2) + e_t^Q\beta_t}{e_t^U\alpha_t + e_t^Q\beta_t} \tag{28}$$

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

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

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

where e_t^U and e_t^Q are the evidence of a return under the distribution of $Q_{h,a}|m, D$ Gimelfarb et al. (2020).

4.4 Max-Boltzmann Exploration

The basic idea of Max-Boltzmann Exploration (MBE) is that of using Softmax policy to explore actions, so as to balance exploration-exploitation trade-off with ϵ-greedy to take advantage of both strategies. This method was proposed by Wiering (1999) and it is based on ϵ-greedy strategy coming with a difference in action sampling in the exploration phase, where actions are sampled over Boltzmann probability distribution, as outlined in Eq. (32).

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

The Max-Boltzmann policy does overcome ϵ-greedy in terms of the exploration probability distribution, as the ϵ-greedy exploration assigns equal probabilities to all non-optimal actions, thereby assigning too large probabilities to explore worse actions Wiering (1999). In contrast to this, by including a softmax policy in the ϵ-greedy, we manage to define a probability distribution that can be far from a uniform distribution for the purpose of avoiding drastically worsening performance. The distance from the uniform distribution is related to the temperature parameter: if τ is too high, 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, introducing the choice of how much to explore.

An improvement of MBE could be drafted by VDBE-Softmax, in which ϵ probability is adjusted with VDBE method, thus resulting in a heuristic strategy based on a data-driven approach Tokic and Palm (2011). The ϵ update is regulated like Eq. (12), totally integrated in MBE framework.

5 Experimental settings and simulation platform

Our study is based on the application of Deep Recurrent Reinforcement Learning techniques in a self-driving car scenario. Particularly, we focused on the analysis of exploration strategies with deterministic and stochastic approaches. Double Dueling Deep Recurrent Q-Learning models have been tested on the AirSim simulator, which provides an environment that can be used to study and test deep learning models of self-driving cars.

All experiments were conducted for 1 million steps on a virtual machine with two GPUs Tesla M60 and Linux OS. AirSim simulator has been connected to Python, in which TensorFlow Compact V2, OpenAI and OpenCV packages were used.

5.1 AirSim simulator: Neighborhood Environment

As previously reported, we used the AirSim simulator to train and test Deep Recurrent Q-Learning models for the purpose of performing autonomous driving tasks. AirSim is an open-source program based on Unreal Engine, which provides some environments. We chose the AirSim NH environment, a small, simplified, urban neighbourhood with a rectangular shape. AirSim NH is composed by a main street connected with side streets which include trees, parked cars, shadows and lights just like in the real world. In fact, Shah et al. (2017) show how AirSim provides realistic environments both physically and visually.

In order to virtualise the simulator, we used OpenGL drivers to perform the AirSimNH file (with .sh extension) and we included one car called PhysicXCar to the environment with one weather condition.

In this case, states consist of environment images extracted from the car front camera, which are cropped in order to highlight the roads. In particular, we defined *Reward* by incorporating a measure of the distance between the position of the car (x_e) and the centre of the roads (x_r), as shown in Eq. (33).

$$r(s, s', a) = e^{-\beta \cdot ||x_e - x_r||} \tag{33}$$

where β is a positive constant and $r \in [0, 1]$ is the reward function.

This formulation, proposed by Riboni et al. (2021), and devised by Spryn et al. (2018), 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.

We defined a discrete set of actions, as shown in the Figure 1, including a center direction and two steering angles for each direction (left and right) under the assumption that the steering angle can be adjusted by a value in the range $[-1, 1]$.

In order to start the processes, the beginning of each episode involves the random extraction of environment coordinates from a set of ten starting points.

5.2 Models experimental setup

As introduced in section 3.1, we considered Double Dueling Deep Recurrent Q-Network model to estimate Q-values for each steering angle and to predict the optimal action based on the input state. We applied the error masking in the equation of the agent update by choosing, differently from Lample and Chaplot (2017), a bootstrapped random update sampling which, as Hausknecht and Stone (2015) cited, represents one of the two kind of sampling available for the recurrent structure.

After reaching a sufficient dimension of the experience buffer, this sampling method requires the construction of experience traces of actions, rewards, current states, next states and terminal nodes at

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 (1)
$target\_Q$ = TN(trace($\mathcal{S}_{t+1}$))                                       ▷ According to (1)
$double\_Q = target\_Q[, \text{argmax}_a\, main\_Q]$                                  ▷ $a \in \mathcal{A}$
$Y = trace(\mathcal{R}) + \gamma \cdot double\_Q \cdot (1 - trace(\mathcal{T}))$
$Q = \max_a$ 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 (2) conditions; $\forall j \in \{1, \ldots, b\}$
```

each epoch. Firstly, a sample of episodes is made by randomly drawing out episodes from the replay memory. Then, each trace of experience could be created by extracting a sub-sequence of steps from each sampled episode, where each initial state is randomly extracted.

By addressing a RL framework, we need to find an estimation of the response variable (Y) in order to estimate the errors of the neural network for updating weights. To deal with this purpose, we use next states sample to estimate two different Q-values ($main\_Q$ and $target\_Q$) by utilizing, respectively, main and target networks and then combining these quantities together with sampled rewards and sampled terminal nodes obtaining Y (see Algorithm 1).

Using a masking error strategy, errors are obtained as a quadratic weighted difference in which the weights assume the 0 or 1 value. The idea behind this strategy is to assign value 0 and value 1 to the initial observations and the final observations of the experience trace, respectively (see Eq. (2)).

In Table 1, all parameter settings of Double Dueling Deep Recurrent Q-Network algorithm[1] are shown.

As reported in Table 1, we assigned two values per buffer size to evaluate how the changing of the experience buffer size might affect the optimization of the neural network. In addition, we chose soft update for Target Network. At each updating step, weights of Target Network are updated with 0.1% of prediction network weights.

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

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*

5.2.1 Exploration strategies

We tested seven different exploration strategies, as reported in the Section 4, with only one setting per strategy. In Table 2, all parametric settings used in the respective fourteen tests are shown.

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

In particular, parameters of BMC method are identical to those considered by Gimelfarb et al. (2020) in their experimental settings. Similarly, parameters of VDBE and VDBE-Softmax are based on experiments carried out by Tokic (2010) and Tokic and Palm (2011).

When we considered an exploration strategy that involves an update regulated by the VDBE equation, the λ parameter was set equal to the inverse of the number of actions and ν equal to a value that was not at the extremes which can be assumed in the range of values, as recommended by the authors.

With Softmax, we assigned a not too high value for κ, even in accordance with the results obtained by Tokic and Palm (2011). For this reason, we extended this value to MBE and VDBE-Softmax.

In a different way, the ϵ value assumed in constant ϵ-greedy is based on the values of ϵ obtained from the results of stochastic methods for ϵ-greedy.

Decreasing ϵ-greedy method required a change in the update structure of D3RQN model, since the equations depend on steps instead of episodes. In fact, in the Table 1 we specified that the starting of the update is fixed to 999 episodes and yet, in this case, we had to set this parameter in steps format because it was incorporated (n_{start}) in the equations that determine the value of ϵ, as specified in Eq. (7) and Eq. (10). We assigned a value of 50000 steps to the n_{start} parameter and 400000 steps for ϵ_{ann}. The other parameters of Decreasing ϵ-greedy strategy were set exactly as the experiments carried out by Riboni et al. (2021).

In addition, we aligned all the methods that include an ϵ management. According to the Decreasing ϵ-greedy strategy, we fixed ϵ equal to 1 to fill the experience buffer until the agent update starts.

6 Results and comparison

In this section, we show the results obtained from the fourteen Double Dueling Deep Recurrent Q-Network models using the AirSim simulator, with the aim of evaluating how seven different exploration strategies and different buffer capacities affect agent learning.

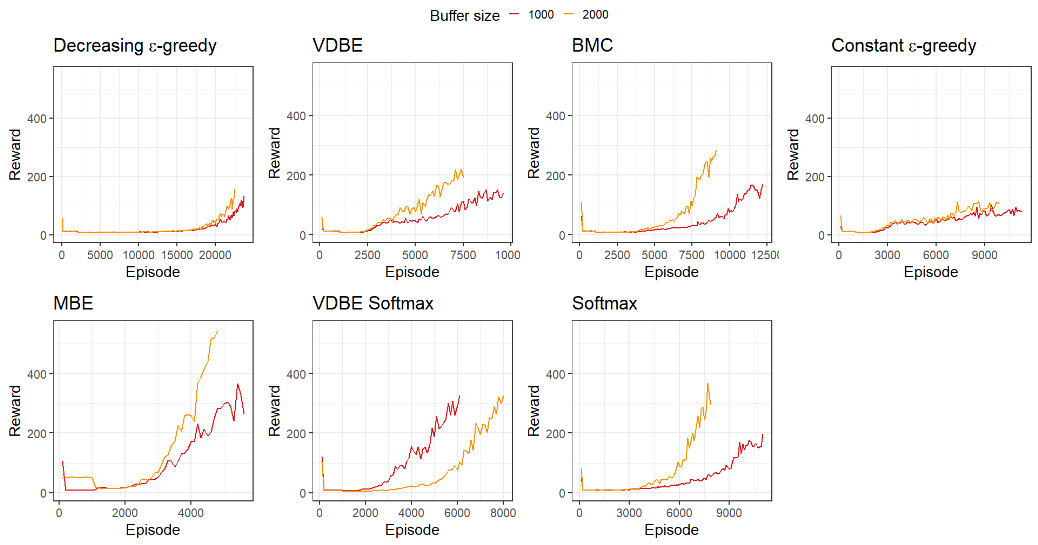
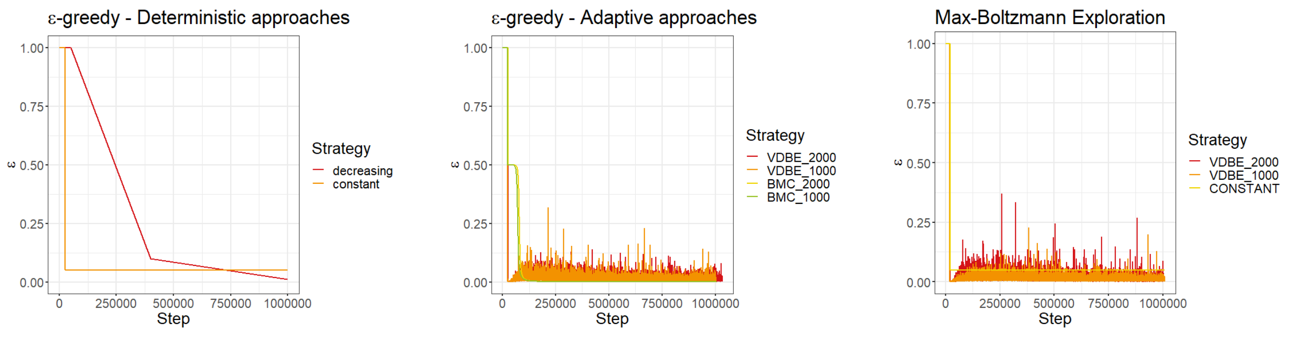


Figure 2: Comparison of the exploration strategies with D3RQN agent. The training curves show the average reward per 100 episodes for each value of the buffer size. The horizontal axis and the vertical axis indicate, respectively, the number of episodes and the average reward.

In Figure 2, Reward trends in the training process are shown. All models show an increasing trend, which means a good policy learning. However, the average reward for deterministic strategies achieves lower values than other strategies.

Another peculiarity which stands out from the data is the difference between ϵ-greedy strategies and the other strategies that involve the Softmax method for exploration: Softmax, MBE and VDBE-Softmax perform better, reaching much higher values in the final steps. This effect may depend on taking a different distribution function for exploration. As reported in the Section 4, 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.



(a) **(b)** **(c)**

Figure 3: 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 steps for adaptive ϵ-greedy strategies (BMC and VDBE) distinguished by buffer size. Figure **(c)** shows ϵ values per steps for Max-Boltzmann Exploration methods (constant and VDBE) distinguished by buffer size.

As it can be seen in Figure 3, stochastic strategies for ϵ-greedy assume 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 is characterised 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 Figure 4, we consider 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 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 buffer of experience. Furthermore, loss differences are too high when the agent improves performance in long-term.

In order to evaluate the goodness of steering angles prediction, we reported a summary of the performance gained from 300 episodes that are evaluated on 10 training starting points and on 10 test starting points respectively. We used different statistical indicators (among others, average,

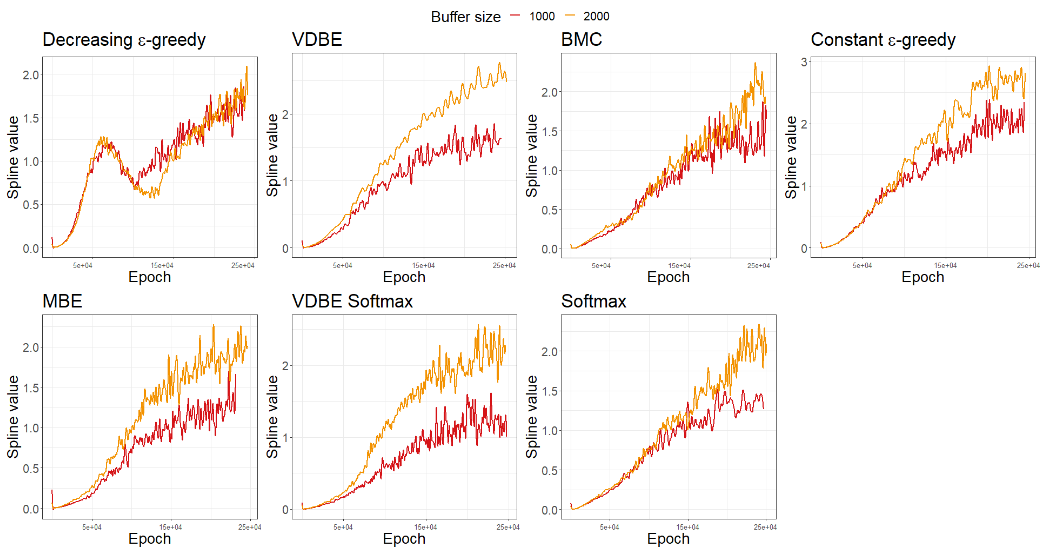


Figure 4: Comparison of training loss differences for each exploration strategy and buffer size. The vertical axis denotes spline values of the absolute difference between the previous and the next loss. The horizontal axis indicates the training epochs.

standard deviation and minimum) of episodes length and Collision-Free Rate (CFR) to evaluate the performance.

Table 3: Training set evaluation. Performance of the agent over the training set with 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%

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 the steps have higher values together with a reduction in episode length variability, especially for the Softmax, VDBE-Softmax, MBE and ϵ-greedy BMC methods.

Table 4: Training set evaluation. Performance of the agent over the training set with 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%

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 than those considered during training.

Table 5: Test set evaluation. Performance of the agent over the test set with buffer size equals 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%

Table 6: Test set evaluation. Performance of the agent over the test set with 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%

We observed that performance still remains very high on the test set, with a tiny discrepancy in comparison with the training set. The only drawback lies in the increasing of steps variability, which may suggest less stability. Generally, Softmax and related methods have better performance than ϵ-greedy strategies, with the exception of the BMC method.

Finally, in Figure 5 we analyzed the optimality related to the action choice in order to understand which model was able to maximise the reward function at each step.

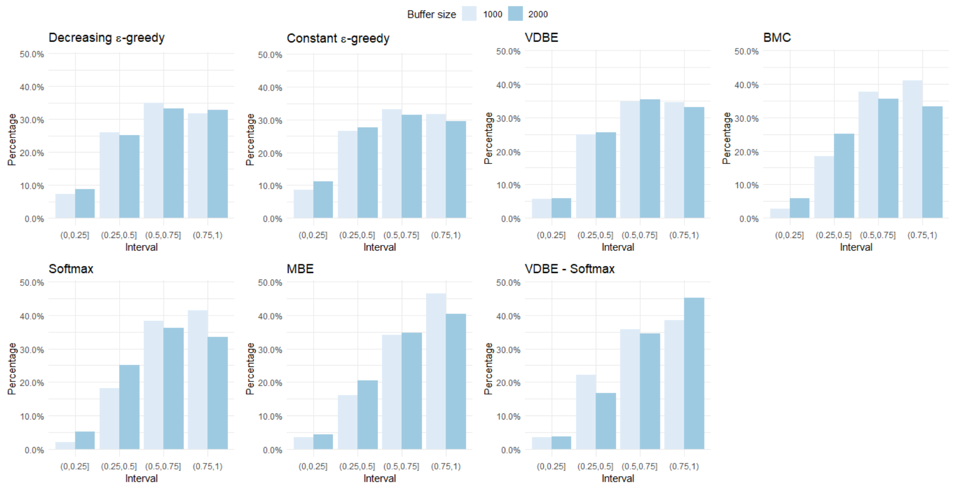


Figure 5: Comparison of test reward values for each exploration strategy and buffer size. Each barplot shows the percentage of reward values falling within each range. The horizontal axis indicates different intervals of reward values in ascending order. The last bin contains the maximum values of reward meaning the optimal choices.

In general, the strategies that achieve the best results above involve more optimal actions. In fact, 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 lasts bins when the buffer size is equal to 1000.

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 and future work

So as to cope with the problems of uncertainty and partial observability when applying Deep Reinforcement Learning to the autonomous driving context, this paper deals with a study of exploration strategies on a recurrent DRL framework. In general, recurrent neural networks bring low learning efficiency in the DRL scenario because of the slow convergence. Firstly, we attempted to solve this issue by combining the D3RQN model, which includes a Convolutional Recurrent Neural Network to predict five steering angle, with an adjustment of the LSTM layer to speed up the learning of neural network in the training process. Subsequently, we evaluated several strategies for exploration to handle the multi-armed bandit dilemma by using both deterministic and stochastic approaches with two classes of methods: ϵ-greedy and Softmax. Softmax and, more generally, adaptive methods are much less studied in the autonomous driving scenario than ϵ-greedy, whose most popular strategy is decreasing ϵ-greedy.

The experimental results demonstrate that, under deterministic approaches, Softmax and MBE achieve better results than ϵ-greedy; however, adaptive strategies better balance exploration by improving collision avoidance. In general, the uniform distribution used to sample actions in ϵ-greedy exploration seems to impair the agent learning in the training process, inducing a worsening of performance. Despite that, using the bayesian model combination to model the updating of ϵ-greedy does ensure good results, even though the estimation of ϵ was close to zero after few epochs and the effect of potential worsening actions which could be obtained by random sampling was reduced. In conclusion, the combination of Boltzmann distribution and ϵ-greedy ensures a better leading of exploration and a better learning from the environment than other methods, thus maximizing reward and achieving more optimal actions.

Future research could continue to explore Max-Boltzmann exploration strategies. As proposed by Gimelfarb et al. (2020), it would be interesting to use Bayesian inference for estimating ϵ probability, as it is the case with ϵ-greedy BMC method.

References

P. Auer, Using Confidence Bounds for Exploitation-Exploration Trade-offs, Journal of Machine Learning Research 3 (2002) 397–422.

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.

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.

M. Gimelfarb, S. Sanner, C.-G. Lee, ϵ-BMC: A Bayesian Ensemble Approach to Epsilon-Greedy Exploration in Model-Free Reinforcement Learning, CoRR (2020).

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, p. 2140–2146.

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.

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.

K. Minwoo, K. Jongyun, J. Minjae, O. Hyondong, Towards monocular vision-based autonomous flight through deep reinforcement learning, Expert Systems with Applications 198 (2022).

C. Núñez-Molina, J. Fernández-Olivares, R. Pérez, Learning to select goals in Automated Planning with Deep-Q Learning, Expert Systems with Applications 202 (2022).

H. Alavizadeh, H. Alavizadeh, J. Jang-Jaccard, Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection, Computers 11 (2022).

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.

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.

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, Applied Energy 254 (2019) 113708.

M. Hausknecht, P. Stone, Deep Recurrent Q-Learning for Partially Observable MDPs, CoRR (2015).

C. Romac, V. Béraud, Deep Recurrent Q-Learning vs Deep Q-Learning on a simple Partially Observable Markov Decision Process with Minecraft, 2019.

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.

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.

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.

Y. Wu, S. Liao, X. Liu, Z. Li, R. Lu, Deep Reinforcement Learning on Autonomous Driving Policy With Auxiliary Critic Network, IEEE Transactions on Neural Networks and Learning Systems (2021) 1–11.

A. Santara, S. Rudra, S. A. Buridi, M. Kaushik, A. Naik, B. Kaul, B. Ravindran, MADRaS: Multi Agent Driving Simulator, Journal of Artificial Intelligence Research 70 (2021) 1517–1555.

Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, A. M. López, Multimodal End-to-End Autonomous Driving, IEEE Transactions on Intelligent Transportation Systems 23 (2022) 537–547.

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.

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.

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.

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.

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.

M. Wiering, M. Van Otterlo, Reinforcement Learning - State of the Art, Springer, 2012.

R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 2014.

R. Dearden, N. Friedman, S. Russell, Bayesian Q-Learning, AAAI/IAAI (1998) 761–768.

M. Wiering, Explorations in Efficient Reinforcement Learning, Ph.D. thesis, University of Amsterdam, 1999.

S. Shah, D. Dey, C. Lovett, A. Kapoor, Aerial Informatics and Robotics Platform, Technical Report MSR-TR-2017-9, Microsoft Research, 2017.

M. Spryn, A. Sharma, D. Parkar, M. Shrimal, Distributed Deep Reinforcement Learning on the Cloud for Autonomous Driving, IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS) (2018) 16–22.