Reinforcement Learning for Truck Eco-Driving: A Serious Game as Driving Assistance System

Mohamed Fassih, Anne-Sophie Capelle-Laizé, Philippe Carré, Pierre-Yves Boisbunon

Reinforcement Learning for truck Eco-driving: a serious game as driving assistance system

Mohamed Fassih[1,2], Anne-Sophie Capelle-Laizé[1], Philippe Carré[1], and Pierre-Yves Boisbunon[2]

[1] XLIM UMR CNRS 7252, Université de Poitiers, 11 Bd Marie et Pierre Curie, 86360 Futuroscope Chasseneuil, France
firstname.lastname@univ-poitiers.fr
[2] Strada, 10 rue Jean Mermoz, 79300, France

Abstract. Making fuel-economy for vehicles is an important and current challenge in particular for professionals of transportation. In this article, we address the challenge of providing a driving serious game based on artificial intelligence in order to significantly reduce the fuel consumption for trunks. Our proposition is based on a machine learning process compound of a Self-organizing Network for clustering task followed by Reinforcement learning process in order to provide accurate recommendations for eco-driving. Driving experts provide us knowledge in order to model the actions-rewards process. Experimentation on simulated data show the recommendations are coherent and allows the drivers to have an eco-driving behavior.

Keywords: Eco-driving · Self-organizing Network · Reinforcement Learning · Serious game

1 Introduction

Eco-driving has become one important aspect in the field of road freight transport. The requirements for adapting an eco-driving behavior are becoming stronger: transport companies aims to reduce their fuel consumption and CO2 emissions for economic and environmental issues. Logistic and transportation enterprises benefit of research in numerical data. Today, most of the trunks are equipped with numerous sensors that provide real-time information. When they are coupled with driving indicators, they can help to punctually optimize driving efficiency. For example, gear shift indicator can be coupled to engine speed.

In this work, we aim to overtake this simple and punctual driving-assistance by a complete eco-driving system which assist the driver in the long term journey. By the analysis of vehicle technical data such as vehicle speed, real-time consumption or braking state our system should be able to evaluate driving quality and propose actions that globally optimize fuel consumption and CO2 emissions. Our solution can be viewed as a serious game where driver has to improve its driving score. This work is jointly developed with the company *Strada*[3]

[3] https://www.stradaworld.com/

specialized in Transport Management System (TMS) for fleet management optimization. *Strada* has a high potential of transport data thanks to there 4800 customers and 5800 connected vehicles. The *Strada* company's objective is to provide drivers wishing to improve their eco-driving awareness and learning tools through a driving simulator.

The analysis of *Strada* requirements involve some assumptions and constraints. First, since it exists several driving modes and situations depending on the drivers, vehicles, journeys, roads, *etc*, the simulator has to self-adapt to the different contexts and has to offer accurate driving recommendations within the context. Secondly, the solution should appears as a positive experience for the learner. Thus a gamification of the solution sounds an appropriate approach. As driving can be viewed as a continuous sequence of actions, our proposition is to develop a simulator that increase the driving performance of drivers in a whole sequence. With these hypotheses reinforcement learning algorithms sound well adapted to our problematic. Indeed, the reinforcement learning (RL) are another path of machine learning approach between unsupervised and supervised learning. The agent learns to behave in environment depending on future rewards, and has a goal to develop efficient policy to optimize a cumulus reward. From our point of view, RL is a adapted response our concern: increase driving score in a gamification approach.

It exists large number of papers about Reinforcement Learning. We can cite the introduction to RL presented by Sutton in [1] and some surveys in [2,3,4]. Obviously, numerous and various practical applications are nowadays address using RL and deep RL. In [5], authors present review of RL for Cybersecurity domain. In [6], healthy problems are treated. In [7], a survey of Deep RL for blockchain in industrial IoT is presented. Closer to our problematic, RL and Deep RL are also used in vehicle management but most of times it concerns autonomous driving context. One survey is proposed by Elallid and al [8]. In particular context of eco-driving, some propositions exists but mainly concerns electric and personal vehicle ([9,10,11]) but trucks consumption assistance in transport of goods still little treated.

In this article, we propose a solution for truck eco-driving. As a chest game, our proposal consists in giving to a driver optimal recommendations (driving actions) that will increase a global performance considering fuel consumption using reinforcement learning algorithm. As RL is based on the principle of action-reward estimation, our solution should be able to identify current state of a vehicle and to provide local rewards. One originality of our proposition is to combine clustering approach for states estimation and RL. Rewards and states are defined within expert knowledge.

The remaining part of this paper is organized as follows. Section 3 dedicate to theoretical aspects of RL and the proposal description: RL basis are first introduced in section 2 and general scheme of our proposal is described in section 3.1. The use of experts' knowledge for actions-rewards modeling is presented in 3.2 and section 3.3 describes how states are estimated. Section 4 presents our

experiments and results. We conclude and propose some new perspectives in section 5.

2 The basis of Reinforcement Learning

Reinforcement Learning is a body of theory and algorithms for optimal decision making developed in the last twenty-five years. RL methods find useful approximate solutions to optimal-control problems that are too large or too ill-defined for classical methods such as dynamic programming. The main explanation and principle can be found in this reference [1] and we review here only the background concepts.

Reinforcement learning is a class of solutions for solving Markov Decision Systems defined such that:

- a set S of states,
- a set A of actions,
- a transition probability function $p : S \times A \rightarrow P_{sa}(.)$ that is the transition probabilities upon taking action a in state s,
- a reward function $R : S \times S \times A \rightarrow r$ which modelizes the reward $R(s_{t+1}, s_t, a_t)$, the expected rewards for state-action-next-state triples,
- a future discount factor γ, the discount rate determines the present value of future rewards.

Markov Decision Processes (MDPs) provide a framework for modeling decision making. The key feature of MDPs is that they follow the Markov Property; all future states are independent of the past given the present. In RL, the goal is to find a policy $\pi : S \rightarrow A$, that maximizes the "action-value function" of every state-action pair, defined as:

$$Q_\pi(s, a) = E_\pi \left[\sum_{t=0}^{T} \gamma^t R(s_{t+1}, s_t, a_t) | a_0 = a, s_0 = s \right] \quad (1)$$

In the previous equation, the expectation, noted E is over the state sequence $(s_0, s_1, ...)$ we pass through when we execute the policy π starting from s_0. In our setting, we use a finite time horizon T (it is a particular context, the end of the truck travel). $Q_\pi(s_0, a_0)$ is the expected cumulative reward received while starting from state s_0 with action a_0 and following policy π. The solution of an MDP is a policy π^* that for every initial state s_0 maximizes the expected cumulative reward.

Reinforcement Learning tests which actions are best for each state of an environment essentially by trial and error. The model sets a random policy to start, and each time one action is taken. This continues until the state being terminal (T in the previous equation).

To solve this problem a classical strategy is to use the Q-Learning algorithm. The Q-Learning algorithm was first introduced by [12], and is one of the most studied methods. Given an MDP, Q-Learning aims to calculate the corresponding

optimal action value function Q^*, and thus we can choose any behavioural policy to gather experience from the environment.

The $Q(s, a)$ function is represented in tabular form, with each state-action pair (s, a) represented discretely [12]. The Q-Learning algorithm converges to an optimal policy by applying the following update rule at each step t:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t, a_t) + \gamma \max_a [Q(s_{t+1}, a)] - Q(s_t, a_t)] \quad (2)$$

$\alpha \in [0, 1]$ is the learning rate. With the ϵ-greedy strategy, the agent choose the optimal action with probability $(1 - \epsilon)$, and to choose a random action with probability ϵ. The value of the parameter can be varied over time, by decreasing it over the course of training.

3 A reinforcement learning solution for eco-driving

In order to use the reinforcement learning for eco-driving it is necessary to be able to define precisely the state space S, the action space A and the reward function. Before detailing these different parameters in our application context, we provides in next section, an overview of the proposed solution for *Strada* eco-driving problematic.

3.1 General proposition for eco-driving simulator

The *Strada* driving experts have identified three distinct phases for heavy-duty vehicles driving: *acceleration* also called start-up phase, *rolling* phase and *braking* phase. These ones will described in the next section. During each of these three phases, the engine characteristics and its performance are very different and the experts estimate that the types of actions a driver can do to improve its eco-driving are significantly different. Given thus point of view, we propose that couples $\{actions - rewards\}$ differ according to the driving phase.

Our global workflow for eco-driving recommendation is described Figure 1. This proposition is divided into 5 stages. The first is the *data collection* which consists of measuring different driving and engine characteristics. The second is a *driving phase detection* in charge of identifying the driving phase of the sequence. The next one consists in *estimating* of the current *state* as an entry of the last stage where the *recommendation* is given. In the next sections, we describe how experts' knowledge can be used in our proposal.

3.2 Actions-Rewards definition using expert knowledge

As previously told, the *Strada* driving experts have identified three distinct situations in a time lapse sequence of driving truck: start-up phase, rolling phase and braking phase a illustrated Figure 2. For each driving sequence, the experts have defined interesting technical parameters to observe in order to assess whether

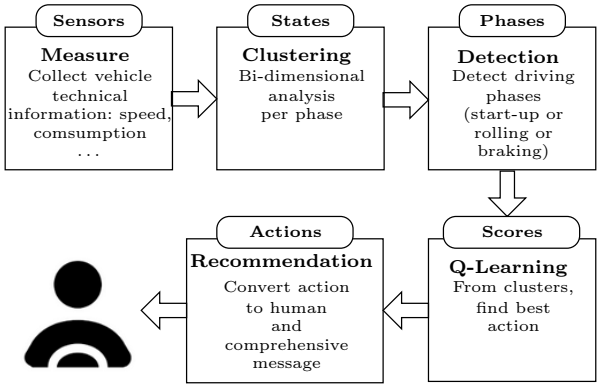

Fig. 1. Global workflow overview

the driver is behaving in the most appropriate way in view of the objective of reducing fuel consumption.

The *Start-up phase* is characterized by a strong speed acceleration in a certain time slot: during this stage, the driver has to increase speed very quickly to reach the cruising speed of the vehicle. Factually, driver can act on the vehicle acceleration (more or less acceleration) and change gears. So, engine speed (in *rpm*) and acceleration (in %) can be used to characterize vehicle state during the *start-up phase*. This couple of measures defines the start-up phase feature space. Using RL, we can consider that each driver's action induces a displacement in this measurement space and we have to be able to associate each position in the feature space with a *state* and a local *score*. *Strada*'s experts, based on their

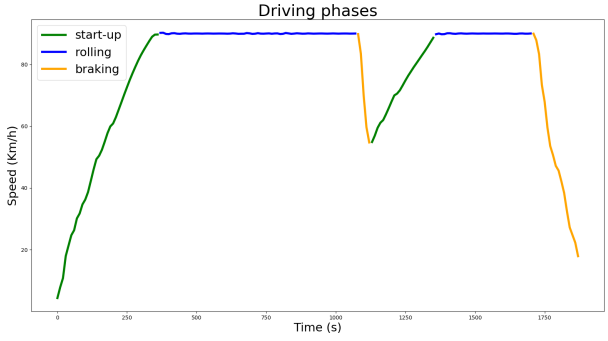

Fig. 2. The three driving phases

experience, propose to divide feature space into several sub-spaces and associate

each sub-space to a local reward. Figure 3 describes their proposal. As we can see, experts estimate a vehicle can be in 12 different states depending on speed and acceleration values. Each state is associated with a reward. The more higher is the reward, the more the engine's characteristics tends near in optimal position (green sub-spaces).

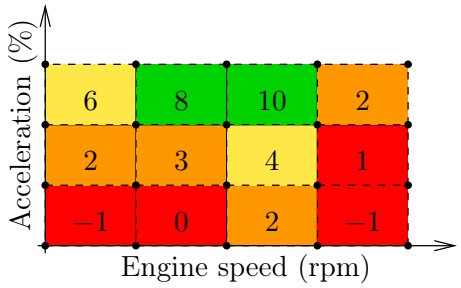

Fig. 3. Start-up phase feature space, states and *scoring* table

On the same principle, scoring tables can be defined for the two others driving phases. Considering that *rolling phase* corresponds to a constant speed without acceleration neither deceleration and *braking phase* corresponds to a strong speed deceleration in a certain time slot, experts propose to retain the measures of variation speed and fuel consumption for the *rolling* phase and the measures of braking percentage and deceleration for the *braking* phase. Figure 4 presents the proposed scoring Tables for these phase and the table 1 summarize measures and driver actions.

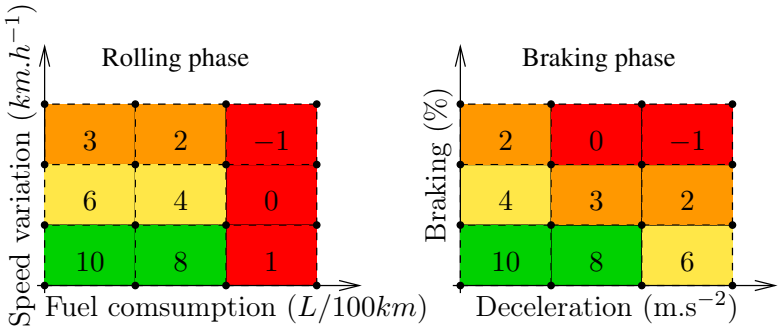

Fig. 4. Rolling and braking phase feature space, states and *scoring* tables

Driving phase	Useful features/measures	Possible driver's actions
Start-up	engine speed, acceleration	increase acceleration, decrease acceleration, change gears
Rolling	speed variation, fuel consumption	use more inertia, use less inertia, use less regulator, use more regulator
Braking	braking percentage, deceleration	less deceleration, more deceleration, use more brake, more deceleration

Table 1. Driving expertise for RL application

3.3 Estimation of the *states*

Figures 3 and 4 define the possible *states* in our RL modeling. The choice by the experts of the number of states and of the values of rewards is based on their experience. It meets their main objective which is the determination of the optimal driving path in the least possible states.

However using *Q-Learning* method for RL implies to be able to identify at each iteration the current state (see eq. 2) of the vehicle. The advantage of the *scoring* tables as defined be the experts is their simplicity. A simple couple of measures provide the state. However, at this point, experts are unable to precisely define the boundaries of each sub-space. Moreover, we have to keep in mind that the boundaries between each sub-space can not be universal. Indeed, many exterior parameters influence driving context (type of journey, cargo, vehicle...) and it seems obvious that the boundaries between states can not be static. They have to be estimated for each driving situation.

Some solutions exist to estimate states by combining the convolutional neural network with the Q-Learning algorithm as proposed in [13]. In this deep Q-Learning [13], a full-connected neural network is used to identify some states according to the input values during a learning state. But this process is very time consuming and requires a large data bases. This due to the fact that during the reinforcement learning algorithm it is necessary to learn the weights associated with the conventional neural network and the full connected part.

In our application, we propose an another strategy: a clustering process in order to identify the state of the vehicle and the sub-spaces boundaries. Our proposition is to use Self-Organizing Map (SOM) which is a well-known unsupervised learning tool. Some authors have already proposed to use SOM in RL [14], [15]. In these articles, the SOM maps the input space in response to the real-valued state information, and each unit is then interpreted as a discrete state. This context is near our problematic. So, we propose to use this structure because its permits one to apply a clustering process that will allow us to detect the states. Moreover, the huge advantage of SOM is its ability to preserves the topological properties of the input space. The number of neurons will correspond to the number of sub-spaces proposed by the experts.

The figure 5 illustrates SOM convergence. Applied to our process, the grid corresponds to the *score* space and blue cloud to real data measures.

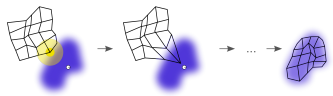

Fig. 5. *2D-SOM* (illustration from wikipedia).

As SOM is a classical algorithm, so we just briefly resume its principles. Native works on SOM can be find in [16,17,18].

The SOM is a prominent unsupervised neural network, considered as a 2D mapping of the data group [18]. SOM net is made up of a number of nodes usually organized in a rectangular grid where an input vector x_i is link to a weigh vector. SOM algorithm is composed of 2 iterative stages: a competitive and a cooperative stage. The competitive stage aims to select the best neuron whereas the cooperative one adjust the weights. Topological aspects are driven during the update using neighborhood function also called kernel. So, in our proposition, the number of neurons equals the number of states. When SOM algorithm has converged, input vectors are labeled. The final position of the neurons are then used to estimate the limits of each subspace as defined by expert. This learning stage thus provides a way to associates a state to each new couple of measures for the Q-learning algorithm.

In that section, we have explained our proposed method based on a first clustering of the driving spaces followed by the RL stage. The next section details experimentation and results of the proposed method.

4 Results of the SOMQL algorithm

In this part, we describe our experimentation, in particular the used data, the clustering stage and the driving recommendations.

4.1 Data generation

Our recommendation system is based on Q-learning algorithm. The convergence of a such system suppose to own a large among of data that covers the representation space: optimal policy is found by navigating through a stochastic maze in the feature space. Obviously, it is not realistic to obtain all the data using a truck inserted in a real traffic. First, some measures could be reached only if driver makes unsuitable actions which may cause damages to trunk. Moreover, some actions would have be dangerous in real traffic conditions. So, we decide to generate simulated data using *Euro Truck Simulator 2* [19] which is a software that proposed various type of truck and journeys (missions). More over, using this simulator, we stay in serious game situation. Figure 6 illustrates a set of collected data for the three specific driving phase.

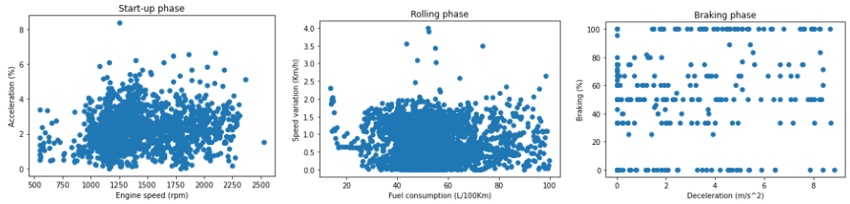

Fig. 6. Example of data corresponding to start-up, rolling and braking phases (from left to right)

4.2 States estimation using SOM

As described in section 3.3, SOM is used in order to estimate the boundaries between the several possible states defined by experts. Indeed, the grids they defined are composed several states/scores but no numerical values were defined. Given the clusters estimates by SOM, one can divide feature spaces into a grid which is specific to a truck type. Figures 7 illustrates the some estimated clustering and grid boundaries. Each subspace can then be associated with experts state and score in reference with the scoring tables presented in Figures 3 and 4).

In the next section, we provide some result obtained by our recommendation system for optimal driving based on this clustering and the QL algorithm.

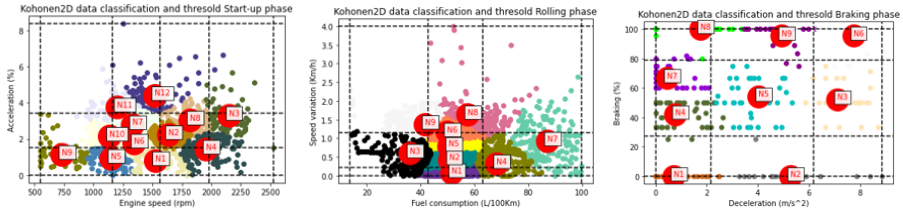

Fig. 7. Kohonen2D result associated with start-up, rolling and braking phases

4.3 Driving recommendation results and discussions

In order to test the recommendation system, we simulate new data for the three specific driving phases and produce recommendations. We here describe and comment some particular cases. First, Table 2 provides some examples of measures obtained during *start-up* phase, their estimated score issued from clustering, and the final recommendation. A graph shows displacements in feature space if drivers apply driving recommendations.

For the *start-up* phase, in case 1 (first row of table 2), the measures correspond to a low engine speed and a low acceleration. In that case, the thresholds learn with SOM indicate that the current state of the vehicle is associated to

local score value of -1 and Q-learning proposes *more acceleration* recommendation. If the driver executes that recommended action, the point will move to the upper state as illustrated with the up arrow on the scatter graph (Figure8, left sub-figure); the new position is closer to ideal position defined by experts.

In the case 2, with a high engine speed and a low acceleration, current state is associated to local -1 score. The system recommends to gear up (*next gear*). This action decreases the engine speed and induces a new position in the feature space by moving to the left. Finally, with case 3, we get a *previous gear* recommendation for a low engine speed and a high acceleration, going to the right of to current state.

Some data measures	**Recommendations**	**Score**
(1) $s = 600$, $a = 0.7$	*more acceleration*	-1
(2) $s = 2650$, $a = 1.1$	*next gear*	-1
(3) $s = 800$, $a = 8$	*previous gear*	6

Table 2. Recommendations for start-up phase (s:engine speed in rmp, a acceleration in m/s^2

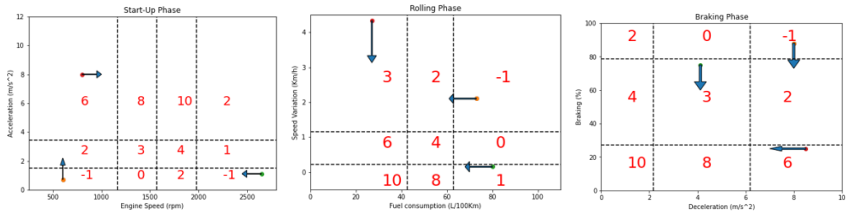

Fig. 8. Displacements in feature/state space - Scatter space (From left to right: for *start-up*, *rolling* and *braking* phases

The same analysis is performed for the rolling and braking phases. The Tables 3 and 4 sum up theses cases. We observe that the system is always trying to reach the high score zone which is the desired behavior for eco-driving optimization. The system seems to be efficient to provide accurate driving recommendations. The recommended actions successively move feature date toward "green" regions of the *score* tables as expected.

Theses results were presented to *Strada* experts to evaluate the global solution on a complete driving sequence. They estimate that the system offers consistent recommendations from an eco-driving behavior point of view. The recommendations proposed by our system seem very relevant. They perfectly meet the driving advises to give according to the simulated driving sequence. Right now, the solution can be viewed of as three parallel processes, one per

Some data measures	Recommendations	Score
$fc = 73$, $\Delta s = 2.1$	*use more inertia*	-1
$fc = 80$, $\Delta s = 0.15$	*use more inertia*	$+1$
$fc = 27.1$, $\Delta s = 4.33$	*use more regulator*	$+3$

Table 3. Recommendations for rolling phase (with fc: fuel consumption in $L/100Km$, Δs: speed variation in Km/h)

Some data measures	Recommendations	Score
$d = 8$, $b = 88\%$	*use less brake*	-1
$d = 4.12$, $b = 75\%$	*use less brake*	$+3$
$d = 8.5$, $b = 25\%$	*less deceleration*	$+6$

Table 4. Recommendations for braking phase (with d: deceleration in m/s^2, b: braking

driving phase. Individually, each process provides accurate recommendations for eco-driving. Obviously, driving a truck is a continuous process where *start-up*, *rolling* and *braking* phases follow one another. Global solution includes automatic detection of the driving phase. Evaluation of global fuel consumption on a simulated travel has also to be done. Given these results, solution will be adapted to real truck. Others search will be engaged based on deep reinforcement learning.

5 Conclusion

In the context of eco-driving, we presented a real-time recommendation system for a simulator-type driving training. The proposed real time recommendations is based on the reinforcement learning. Since the states domain is continuous, we introduce an identification of discrete states by using a Self-organizing Map. All the parameters of our strategy are setting by qualified expert. Finally, we use our proposed system on simulated driving, and according to the expert, the recommendations are coherent and allows the driver to have an eco-driving behavior. Perspectives of this work are numerous. First, introducing Deep RL should simplify identifications of the current states for Q-learning. However, as each system based on deep approach, large amount of data still a requirement. It should be difficulty with real data in transportation context. Secondly, the proposal can be enriched by taking into account new information such as type of road during the travel (streets, roundabout, highway...), topology of missions or shipments.

Acknowledgments

This work was supported by the ANRT funding. Special thanks to the Strada team for their time, knowledge and expertise.

References

1. R. S. Sutton, A. G. Barto, Introduction to Reinforcement Learning, 1st Edition, MIT Press, Cambridge, MA, USA, 1998.
2. M. Wiering, M. Van Otterlo, Reinforcement Learning: State of the Art, Springer, 2012.
3. J. Garcıa, F. Fernández, A comprehensive survey on safe reinforcement learning, Journal of Machine Learning Research 16 (1) (2015) 1437–1480.
4. L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, Journal of artificial intelligence research 4 (1996) 237–285.
5. A. M. K. Adawadkar, N. Kulkarni, Cyber-security and reinforcement learning — A brief survey, Engineering Applications of Artificial Intelligence 114 (2022) 105116. doi:10.1016/j.engappai.2022.105116.
6. A. Coronato, M. Naeem, G. De Pietro, G. Paragliola, Reinforcement learning for intelligent healthcare applications: A survey, Artificial Intelligence in Medicine 109 (2020) 101964. doi:10.1016/j.artmed.2020.101964.
7. M. S. Frikha, S. M. Gammar, A. Lahmadi, L. Andrey, Reinforcement and deep reinforcement learning for wireless Internet of Things: A survey, Computer Communications 178 (2021) 98–113. doi:10.1016/j.comcom.2021.07.014.
8. B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, N. Mrani, A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving, Journal of King Saud University - Computer and Information Sciences 34 (9) (2022) 7366–7390. doi:10.1016/j.jksuci.2022.03.013.
9. K. Yeom, Model predictive control and deep reinforcement learning based energy efficient eco-driving for battery electric vehicles, Energy Reports 8 (2022) 34–42. doi:10.1016/j.egyr.2022.10.040.
10. J. Li, X. Wu, M. Xu, Y. Liu, Deep reinforcement learning and reward shaping based eco-driving control for automated HEVs among signalized intersections, Energy 251 (2022) 123924. doi:10.1016/j.energy.2022.123924.
11. G. Du, Y. Zou, X. Zhang, T. Liu, J. Wu, D. He, Deep reinforcement learning based energy management for a hybrid electric vehicle, Energy 201 (2020) 117591. doi:10.1016/j.energy.2020.117591.
12. C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.
13. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. A. Riedmiller, Playing atari with deep reinforcement learning, CoRR abs/1312.5602 (2013).
14. Y. Osana, Reinforcement learning using kohonen feature map probabilistic associative memory based on weights distribution, in: Advances in Reinforcement Learning, IntechOpen, 2011.
15. H. Montazeri, S. Moradi, R. Safabakhsh, Continuous state/action reinforcement learning: A growing self-organizing map approach, Neurocomputing 74 (7) (2011) 1069–1082.
16. T. Kohonen, Self-organized formation of topologically correct feature maps, Biological cybernetics 43 (1) (1982) 59–69.
17. T. Kohonen, Self-organization and associative memory, Vol. 8, Springer Science & Business Media, 2012.
18. T. Kohonen, Essentials of the self-organizing map, Neural Networks 37 (2013) 52–65, publisher: Pergamon. doi:10.1016/j.neunet.2012.09.018.
19. Euro Truck Simulator 2.
URL https://eurotrucksimulator2.com/