Bulletin of the *Transilvania* University of Braşov • Vol. 10 (59) No. 2 - 2017
Series I: Engineering Sciences

DEEP REINFORCEMENT LEARNING FOR AUTONOMOUS VEHICLES - STATE OF THE ART

L. MARINA[1] A. SANDU[1]

Abstract: *Reinforcement learning is considered to be one of the strongest paradigms in AI domain, which can be applied to teach machines how to behave through environment interaction. Recently the concept of deep reinforcement learning (DRL) was introduced and was tested with success in games like Atari 2600 or Go, proving the capability to learn a good representation of the environment. In this moment there are few implementations of DRL in the autonomous driving field. In this paper, we present the state of the art in deep reinforcement learning paradigm highlighting the current achievements for autonomous driving vehicles.*

Key words: *reinforcement learning, deep neural networks, autonomous driving.*

1. Introduction

An autonomous vehicle, capable of driving through various environments and to learn from unexpected situations, is one of the most important goals of AI. Driving a vehicle represents a complex task even for a human driver, thus autonomous driving represents a very difficult challenge, in terms of reliable and safety decisions that should be taken.

Nowadays, due to the computation capabilities, deep neural networks are used to learn successful policies directly from high-dimensional sensor inputs. Convolutional neural networks (CNN) are used with success for traffic scene perception, being capable of object recognition, like pedestrian or vehicles, lane detection and also distance estimation.

Currently, autonomous driving implementations are not based on computer vision techniques because of a lack of robustness. One of the most difficult challenges is to compress the input image into a representative feature vector. In this moment there are two approaches used to solve this problem: “mediated perception approaches” and ”behavior reflex approaches” [4], [19].

Mediated perception approach parse the entire traffic scene which is represented as an input image and analyze it involving multiple sub-components for specific object recognition, like traffic signs, lanes, pedestrian, vehicles. Using the recognition results, a consistent world representation can be created and a machine learning (ML) based engine will take into account all the information for decision making. The task of scene understanding adds more complexity to neural networks (NN) algorithms and leads to

[1] Dept. of Automation, *Transilvania* University of Brașov.

performances issues. To eliminate this disadvantage instead of detection a bounding box of an object, a distance prediction to the objects can be done [4].

Behavior reflex approaches utilize a regression to directly map sensors inputs to driving actions. This approach was published several years ago, back to the beginning of the 1990s, when [12] used an NN to create a map from an image to a steering angle. This solution is implemented also by NVidia in [3] where CNN's were chosen to map raw pixels from a single front-facing camera directly to steering commands. Known also as an end-to-end approach, this solution proved surprisingly powerful. To train a model, a human driver drives the car along the road, in different traffic scenarios. The system records the images from the mounted camera and also the steering commands. With minimum training data from humans, the system is able to learn how to drive in traffic on local roads with or without lane markings and on highways. Using this approach the system is able to operate in areas with unclear visual guidance, like in parking lots.

The methods described above are based on supervised learning techniques that imply a big amount of data for training sessions. To have access to sufficient and complete datasets represents an impediment for the majority of researchers which are developing deep learning algorithms.

Another approach for autonomous driving implementation is deep reinforcement learning (DRL), a concept that was first introduced at the end of 2013 by a small company from London, named DeepMind [10]. They demonstrated how this concept can be used to train an algorithm to play Atari 2600 video games by observing just the screen pixels and receiving a reward calculated taking into consideration the game score. The result was surprisingly good, taking in consideration that the games were different, with goals designed to be challenging for humans. Using the same model architecture, seven games were learned and three of them performed better than humans. The success of this algorithm, also known as Deep Q-network (DQN) [11], is due to the used architecture which combines two paradigms: reinforcement learning (RL) and deep learning (DL).

In this article, we present the current state of the art in DRL. We want to present the current status, the related work and also the advantages and disadvantages in using the DRL concept. The concepts of RL and DRL are briefly described and the future work regarding applying DLR algorithms for a simulated autonomous car is presented.

2. Related Work

The objectives of this work are to present the current status of the research for deep reinforcement paradigm, concentrating more on the applicability in autonomous vehicles field. We briefly introduced the relevant papers which demonstrate the applicability of DQN algorithms for various applications and we have presented the RL and DRL methodologies.

Deep reinforcement algorithms were first used to train artificial intelligence agents for playing seven Atari 2600 video games [10]. Due to the resounding results achieved using deep Q-learning algorithms the research community started to improve these algorithms in order to use them in other domains, like autonomous driving.

DeepMind company continued the research started in [10] by improving the DQN algorithms in [11], adding a technique named experienced replay in which the agent experience at each time-step is stored into a replay memory. The algorithm stores the last *N* tuples in the replay memory and randomly samples uniformly when performing updates. In this way, the updates variance is reduced comparing with learning from consecutive samples. Another improvement presented in this work was to use a separate network for generating the targets y_j, cloning the network Q to obtain a target network $\hat{Q}$. This modification makes the algorithm more stable compared with the standard online Q-learning. This new approach was applied with success for 49 Atari 2600 games. Even if every game is different, having different challenging tasks and objectives, the implemented DQN algorithm was capable of achieving human-level performance for most of the games and for some of them, even better performances. One of the most important achievements was the mastering of game Go with deep neural networks [14]. In this work, the neural networks were trained with a combination of supervised learning from human experts and reinforcement learning. The algorithm, named *AlphaGo*, achieved a 99.8% average winning rate against other Go programs and defeated the European Go champion by 5 games to 0.

Another interesting and innovative approach to deep reinforcement learning was the work introduced in [20], where the concept of dueling network architectures for DRL was defined. A dueling network represents two separate estimators: one for the state value function and one for the state-dependent action advantage function.

Starting from these implementations, Q-learning algorithms for autonomous driving were developed. In this moment there are not many implementations for autonomous vehicles using DRL. One of the first implementations was presented in [19], where a simulated autonomous vehicle was controlled using DQN methodology. A simple simulated racing game, written in JavaScript, was used to train an agent to control the simulated car. The agent successfully learned the turning operations after the training process, being able to navigate larger sections of the simulated sections without any crash. The algorithms were updated to adapt an infinitely larger state space and to learn an action-value function that provides the action that should be taken in a given state.

Another approach for autonomous driving was recently published in [9], where an agent was created to perform the task of an autonomous car driving from raw sensor inputs. Using Keras DL framework and Vdrift (http://vdrift.net/), an open-source, cross-platform racing game, as the simulation environment, the designed DQN agent was able to navigate on specific tracks.

A DRL framework was proposed also in [1], where TORCS, another open source game simulator, was used to train an agent to plan the sequence of driving giving the surrounding environmental conditions. The particularity of this paper is that the inputs for DQN algorithms are the environment states; instead of images raw sensor inputs are used, aggregated over time, and the output is the driving action. To achieve the space aggregation, two NN are necessary, one for sensor fusion and another for space features.

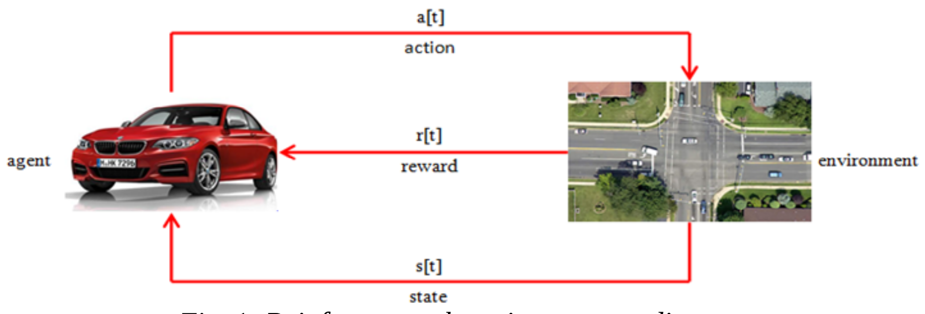

Fig. 1. *Reinforcement learning concept diagram*

The paper with the most significant result is [8], where the success of Deep Q-learning was adapted to continuous actions domains. An actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces was introduced. TORCS was also used as simulation environment for autonomous driving, where the actions are acceleration, braking, and steering. The reward function provides a positive reward at each step for the velocity of the car projected along the track direction and a penalty of -1 for collisions. The algorithm developed is based on DPG algorithm [15] and the critic $Q(s, a)$ is learned using the Bellman equation as in Q-learning.

3. Background

Reinforcement learning (RL) is an area of ML which was formulated in [16] as a model to provide the best policy an agent can follow. Software agents are concerned with taking actions in an environment so that the cumulative reward to be maximized. In Figure 1 can be observed a simplified version of RL concept that can be used to understand a traffic scene. Several successful implementations using RL were done in real time control systems like dynamic robot systems for manipulation and autonomous driving [7].

RL has two main threads: learning by trial and error and the problem of optimal control. The term of optimal control came into use in the 1950s and deals with the problem of controlling a system to achieve certain optimality criterion. One of the approaches to this problem was developed by Richard Bellman, using the concepts of a dynamical system's state and an optimal return function. Also, Bellman introduced the discrete stochastic version of the optimal control problem known as Markovian decision processes. If the probabilistic behavior of a process in the future depends only on its present value, not on the sequence of events that preceded it has the Markov property. An RL task that satisfies the Markov property is called a Markov decision process (MDP), which comprises of a state space S, an action space A, a discount factor γ, a stationary transition dynamics model with density P than satisfies the Markov property $p(s_{t+1}|s_1, a_1, ...s_t, a_t)$ and a reward function $R(s_t, a_t) : S \times A \to \Re$ [2]. Taking into consideration the information stated above, MDP can be defined as a tuple of (S, A, P, R, γ). The formal description of a basic RL problem is a Markov decision processes.

To evaluate *how good* it is for the agent to be in a given state (or how good it is to perform a given action in a given state), value functions are used. The notion of "*how good*" here is defined in terms of future rewards that can be expected [17]. The value of a state s under a policy π denoted as $v_\pi(s)$ can be define using the Equation:

$$v_\pi(s) = \mathrm{E}_\pi[G_t|S_t = s] = \mathrm{E}_n\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \tag{1}$$

where G_t is the gain signal received from the environment. The agent scope is to maximize the gain, which in the simplest cases can be defined as a sum of rewards.

Similarly, the equation for the action-value a in state s, denoted as $q_\pi(s, a)$ can be defined:

$$q_\pi(s, a) = \mathrm{E}_\pi[G_t|S_t = s, A_t = a] = \mathrm{E}_n\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]. \tag{2}$$

An important feature for the equations described above is the possibility to describe it using a recursive representation, where the state value depends on next state value. Resulted Equations are known as Bellman equations:

$$v_\pi(s) = \mathrm{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right], \tag{3}$$

$$q_\pi(s,a) = \mathrm{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]. \tag{4}$$

One of the most used algorithms to solve MPD problems is Q-learning, based on the Bellman equation. The actions are obtained for every state, based on the action-value function:

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\left[R + \gamma \arg\max_{a'} Q_t(s',a')Q_t(s,a)\right]. \tag{5}$$

The algorithm starts from an initial state, and continue until the episode ends. In each step, the agent which is in the current state s, it takes an action following the policy $p(s)$ and then observes the next environment state s' together with the reward R. This algorithm in mainly used in DRL paradigm.

4. Deep Reinforcement Learning Methodology

Using RL in a situation approaching real-world complexity leads to tasks difficult to manage by agents; they must derive an efficient representation of the environment from high-dimensional sensory inputs. RL achieved success in a multitude of domains [13], [5] but the algorithms are limited to domains in which the features can be handcrafted or with fully observable state spaces. Recently, important progress has been made by combining RL with DL concept, creating an algorithm named “Deep Q Network” (DQN) [11] which, using unprocessed pixels from the input is capable of human level performances on several Atari video games. Deep CNNs were used to approximate the action-value function (5) using high dimensional images as states. In Figure 2 an example of deep q-network architecture is presented. The network consists of three convolutional layers and two fully connected layers. Even if the advantages of using deep neural networks, comparing to the classical neural networks, are not clear defined, DNN were chose due to their capability to use graphical computing units for faster computation and also for some algorithmic improvements like drop out, regularization and parameter sharing (e.g. in convolutional layers).

RL is considered to be unstable when using nonlinear approximators [18], such as DNN, because of the correlations presented in the sequence of observations and also because a small change of the action-value function (also known as Q function) may significantly change the policy and change the data distribution. In order to solve this inconvenience, a method of asynchronous training for Q-networks was released [11], a method called experience replay. The experience $e = \{s_t, a_t, r_t, s_{t+1}\}$, also known as transition [19], is stored in a buffer and is accessed uniformly during the training session. This approach leads to an optimization of the loss function, in which θ are the parameters (that are, weights) of the Q-network at iteration i.

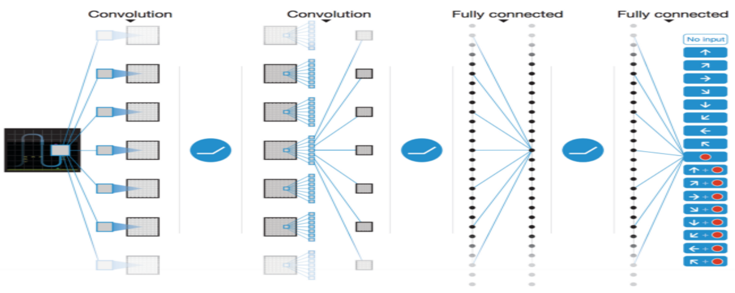

Fig. 2. *Example of Deep Q-Network used in* [11]

$$L(\theta) = \mathrm{E}_{(s,a,r,s')}[(r + \gamma \max_a Q(s', a \mid \theta^-) - Q(s, a \mid \theta)^2]. \tag{6}$$

A second regularization technique applied also in [11] is to use two CNN. The second network proposed, is the "target" Q-network ($\hat{Q}$), which generates target value for the network' loss function. The target network is updated after a specific number of steps. The loss function becomes:

$$L(\theta) = \mathrm{E}_{(s,a,r,s')}[(r + \gamma \max_a \hat{Q}(s', a \mid \theta^-) - Q(s, a \mid \theta)^2]. \tag{7}$$

4.1. Double Q-Learning (DDQN)

To control an autonomous vehicle, an implementation of Double Q-Learning concept can be used [6]. Also known as Double deep Q-networks, this algorithm is an improved version of DQN, which is meant to reduce the DQNs problem of overestimating the value of an action in some situations by separating the behavior policy and the evaluation of actions. Double DQN replaces the original target y_i evaluation function:

$$y_j = r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-), \tag{8}$$

with:

$$y_j = r_j + \gamma \hat{Q}(\phi_{j+1}, \arg\max_a Q(\phi_{j+1}, a; \theta), \theta^-). \tag{9}$$

4.2. Deep Dueling Network Architecture

Dueling architecture is an algorithm described in [20] that separates the representation of state values and action advantages. The network architecture consists of two streams that represent the value and advantages functions and shares a common convolutional learning feature module. The architecture can be seen in Figure 3. These two streams are aggregated via a specially designed layer and produce an estimation of the action-value function. This network, with two streams, is replacing the single stream Q-network which is used in the algorithms described above. Advantage updating was shown to converge faster that Q-learning in simple continuous time domains. The dueling architecture presents both the value $V(s)$ and the advantages $A(s, a)$ using a single deep model, whose outputs combines these two functions to calculate the state action value $Q(s, a)$. Because the output is the same like in the case DQN or DDQN, this network can be trained by any value iteration method.

Considering the stream outputs as $V(s \mid \theta, \beta)$ and $A(s, a \mid \theta, \alpha)$, with θ and α as convolutional network parameters, the aggregation module is constructed as follows:

$$Q(s,a \mid \theta,\alpha,\beta) = V(s \mid \theta,\beta) + \left(A(s,a \mid \theta,\alpha) - \frac{1}{\|A\|}\sum_{a} A(s,a \mid \theta,\alpha)\right). \tag{10}$$

Deep dueling network architecture was used to train an agent to learn 57 Atari games from raw pixels observations. The performance was 75% better comparing with Q-networks, as is stated in [20]. This model became the state of the art for reinforcement learning. Until now, it was not applied for autonomous driving applications but can represent a solution for further developments.

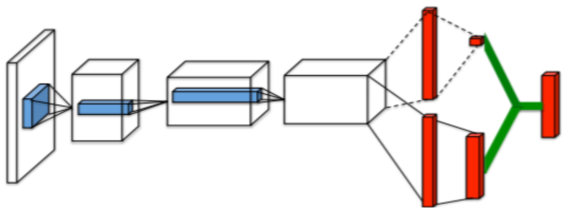
Fig. 3. *Deep dueling network architecture with dual streams to estimate value and advantage function*

5. Conclusion

Deep reinforcement learning paradigm can be used with success in various environments, developing agents capable of dealing with different tasks. As it was presented in this paper, DQN algorithms can be used for simple games, like Atari 2600, for more challenging tasks like *AlphaGo* and also in controlling of autonomous vehicles. We believe that DRL approach is a solution the existing problems in the autonomous driving field. The algorithms are capable of improving the decisions, learning from the unexpected situations. Even if in the automotive industry, supervised learning is the preferred, due to the safety regulation, the current advances in deep Q-learning prove that this concept can represent a solution for an autonomous vehicle in the future.

6. Future Work

The next steps in our research will be to develop an AI agent, reimplementing the algorithms stated in [19], using TORCS, an open source car racing game, as a simulation environment. Our research is done in the autonomous driving field, the scope is to deploy the DQN algorithms on a model car, to see how will behave in a real-world situation. The algorithm will use a pre-trained deep neural network model, the scope being not to train an agent from scratch, but to its improve the behavior.

Acknowledgment

We hereby acknowledge the structural funds project PRO-DD (POS-CCE, O.2.2.1., ID 123, SMIS 2637, ctr. No 11/2009) for providing the infrastructure used in this work.

References

1. Ahmad El Sallab, Mohammed Abdou, Etienne Perot, Senthil Yogamani: *Deep Reinforcement Learning framework for Autonomous Driving.* In: NIPS, 2016.
2. Amarjyoti, S.: *Deep Reinforcement Learning for Robotic Manipulation-The State Of The Art.* In eprint arXiv:1701.08878.
3. Bojarski, M., Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P.: *End to End Learning for Self-Driving Car.* In: arXiv, 2016.
4. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: *DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving.* In: IEEE CVRP, 2015.
5. Diuk, C., Cohen, A., Littman, M.L: *An Object-Oriented Representation for Efficient Reinforcement Learning.* In: Proc. Int. Conf. Mach. Learn., (2008), p. 240-247.
6. van Hasselt, H., Guez, A., Silver, D.: *Deep Reinforcement Learning with Double Q-learning.* In: arXiv:1509.06461v3 [cs.LG].
7. Kober, J., Bagnell, J.A., Peters, J.: *Reinforcement Learning in Robotics: A Survey.* In: The International Journal of Robotics Research, 2013.
8. Lillicrap, T.P., Hunt, J., Pritzel, A., et al.: *Continuos control with deep reinforcement learning.* In: arXiv: 1509.02971.
9. Matt, V., Aran, N.: *Deep Reinforcement Learning Approach to Autonomous Driving.* In: arXiv, 2017.
10. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D.: *Playing Atari with Deep Reinforcement Learning.* In: arXiv preprint arXiv:1312.5602.
11. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: *Human-Level Control Through Deep Reinforcement Learning.* In: Nature, 2015.
12. Pomerleau, D.A.: *Neural Network Perception for Mobile Robot Guidance.* In: Technical report, DTIC Document, 1992.
13. Riedmiller, M., Gabel, T., Hafner, R., Lange, S: *Reinforcement Learning for Robot Soccer.* In: Auton. Robots **27** (2009), p. 55-73.
14. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Driessche, G.: *Mastering the Game of Go with Deep Neural Networks and Tree Search.* In: Nature, 2016.
15. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: *Deterministic Policy Gradient Algorithm:* In: ICML, 2014.
16. Sutton, R.S..: *Learning to Predict by the Methods of Temporal Differences.* In: Machine Learning **3** (**1**) (1988), p. 9-44.
17. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction.* Second Edition, The MIT Press, 2012.
18. Tsitsiklis, J., Roy, B.V: *An Analysis of Temporal-Difference Learning with Function Approximation.* In: IEEE Trans. Automat. Contr. **42** (1997), p. 674-690.
19. Yu, A., Palefsky-Smith, R., Bedi, R.: *Deep Reinforcement Learning for Simulated Autonomous Vehicle Control.* Stanford University.
20. Wang, Z., de Freitas, N., Lanctot, M.: *Dueling Network Architectures for Deep Reinforcement Learning.* In: arXiv preprint arXiv: 1511.06581,2015.