A Hybrid Machine Learning-based Control Strategy for Autonomous Driving Optimization

Ahmad Reda, Rabab Benotsmane, Ahmed Bouzid, József Vásárhelyi

Institute of Automation and Info-communication, University of Miskolc, Egyetemváros, 3515 Miskolc, Hungary, {autareda, iitrabab, ahmed.bouzid, vajo}@uni-miskolc.hu

Abstract: Developing autonomous vehicles is a highly important topic in the field of intelligent transportation systems. Automated steering is a crucial function in the autonomous vehicle. Therefore, it is urgent to either develop a new effective control strategy or improve existing ones. A variety of control strategies are used for this purpose, most with limitations related to their computing capabilities with the highly complex systems or to lack of efficacy related to maintaining the balance between driving performance and driving smoothness. In this paper, three different machine learning-based models were developed to perform an autonomous driving task: a supervised learning model (Deep Neural Network, DNN), a reinforcement Deep Q-learning model (DQN), and a hybrid model. The DNN model was trained based on the behavior of the classical MPC controller. The DQN was designed with the same structure as the DNN and trained by directly interacting with the driving environment. The hybrid model is a combination of supervised and reinforcement learning algorithms, where the trained DNN model is used as a decision-maker (Actor) in a deep deterministic policy gradient reinforcement learning model. The behavior of the designed models was compared based on several performance indicators, including the ability to drive the vehicle along the desired trajectory, the response time, and the smoothness of the driving system. The results show that the DNN model was able to imitate the behavior of the traditional MP Controller efficiently and all three machine learning models successfully drive the vehicle along the desired path. The hybrid model achieves the best results and improved the smoothness of the driving system with a reasonable response time.

Keywords: Autonomous Driving; Model Predictive Control (MPC); Supervised Learning; Deep Neural Networks; Reinforcement Learning; Deep Q-Network (DQN); Deep Deterministic Policy Gradients (DDPG)

1 Introduction

The evolution of autonomous driving systems has seen the use of different technologies aiming to improve efficiency, enhance driving safety and reduce the risks related to traffic congestion. Driving in a structured environment and highway

driving projects were some of the earliest autonomous vehicle projects, carried out at Carnegie Mellon University and Bundeswehr University Munich [2], [3]. Since then, projects and research related to autonomous vehicles have been carried out by academic institutes and companies alike. According to the Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems "SAE-J3016", vehicle autonomy is divided into six different levels. Level 0 (No Automation) depends on the human driver to perform all the driving tasks, it is manually controlled. Level 1 (Driver Assistance) is considered the lowest automation level, where the driver has full responsibility, but some assistant driving systems are included for certain circumstances. Level 2 (Partial Automation) combines different automated functions which can be working simultaneously, such as steering and acceleration tasks, but the driver is still involved in the driving tasks such as performing the maneuvers and has to monitor the environment all the time. At Level 3 (Conditional Automation) the vehicle has the capability of detecting the surrounding environment and making decisions in normal conditions, but the necessity of the driver still exists, meaning that the driver has to be ready to take control over the vehicle at any time. At Level 4 (High Automation) the vehicle performs all the driving tasks in most circumstances, and the driver still has the option to take control. At Level 5 (Full Automation) the vehicle is capable of performing all driving tasks in all circumstances, and the driver has the option to manually override [4], [5]. The vehicle interacts with the surrounding environment in order to perform several related tasks: perception, where the required information about the driving environment is provided to the system; planning, where the optimal scenarios and the control actions are obtained based on the provided information; and the control function, where the control strategy is put into action [6]. The automated steering task is a part of the control function, where the tracking errors are minimized in order to follow the desired trajectory. Driving the vehicle along the desired trajectory is considered one of the most critical tasks due to the fact that any failure in the applied control strategy can have severe consequences. A variety of control strategies have been used to perform the automated steering task, such as the classical feedback control algorithm, Model-Based Control, Dynamic Control, and Adaptive Control [7], [8], [9], [10]. In this context, Model Predictive Control (MPC) has become the most commonly used algorithm for the autonomous vehicle steering system. The MPC controller solves an online optimization problem with the ability to handle the system constraints (soft-hard) by including them in the design process, which makes it a powerful strategy to deal with the stability and the changing dynamics of the vehicle. On the other hand, with the increase of the system complexity, the computational load of the MPC controller is increased, since it solves the optimization problem in each time step, and it may not be able to meet the real-time requirements. Additionally, MPC is resource-consuming, which makes it invisible, especially when it comes to the limited resources of embedded computing platforms such as system-on-chip (SoC) and field-programmable gate array (FPGA) adaptive platforms [11], [12]. Recently, Deep Neural Network (DNN) has gained attention and has been rapidly developed

and efficiently implemented with a variety of applications in different fields such as image classification [14], natural language processing, and speech recognition [15]. In contrast, to the classical control algorithms, which are mainly based on tuning predefined parameters related to a determined environment [16], the behavior of the deep neural network model is optimized based on the provided information (self-optimized algorithm). In other words, the neural networks algorithm bypasses the need for significant parameter tuning, which makes it more efficient to model highly complex systems and to deal with unforeseen situations, especially after being well trained and validated using sufficient datasets. Recently the implementation of deep neural networks within the domain of robotic applications has made massive progress and has provided promising results such as perception and motion planning [17] and object detection and semantic segmentation [18]. In contrast, to supervised learning, agents in Reinforcement Learning (RL) are trained by directly interacting with their environment rather than explicitly guiding the model on how to act based on the labeled data [19]. The performance of the RL agent is evaluated based on the reward function, where the agent is trained to act in the environment in a way that maximizes the cumulative reward in order to improve the performance [20]. RL has proven to be a powerful method mainly in the domains of game playing and robotic manipulation [21], [22], and RL algorithms are considered a promising potential solution for many other applications, especially in cases where classical supervised learning is not applicable. Although there are promising results achieved by the implementations of reinforcement learning with different complex tasks related to automated driving, RL is still an emergent field in this domain, where the implementations and deployment of real-world applications are still very much an open challenge and RL has not yet been applied to practice as successfully as supervised and unsupervised learning. The main contributions of this work can be summarized in two main points. The first is leveraging the advantages of reinforcement learning and supervised learning by combining them in one control model in such a way that the RL-based network optimizes the action that is taken by the supervised neural network (DNN) and achieves a better generalization capability with the complex driving environment. The second contribution comes in enriching the research on RL algorithms and paving the way to bring RL closer to real-world implementations. In [13], a classic MPC controller was designed and deployed on FPGA for automated driving task, while in this paper three different machine learning-based models are developed for the same task and compared to the traditional MPC. The first model is a DNN-based model, which is designed and trained using a supervised dataset obtained from the behavior of the classical MPC controller. The second model is a reinforcement learning-based model (DQN) which is designed and trained without any supervision data, but directly by interaction with the environment. The third model is a hybrid one, which is a combination between the DNN and reinforcement learning methods. The trained DNN will be used as decision maker working beside another network (critic) within a DDPG reinforcement model. The combined method is expected to provide an

optimized solution, as the actions that are taken by the decision maker (trained DNN) will be evaluated and optimized by another neural network in order to minimize errors. Additionally, the combined model will be able to deal with and adapt to new cases that have not been faced during training.

The paper is organized and structured as follows: The second section provides background, including the most common vehicle models and control strategies that are used for autonomous driving, in addition to the work related machine learning algorithms describing the main features and their implementations in the field of autonomous driving. In the third section, the MPC controller and the design of the suggested models are discussed. The implementations and the obtained results are analyzed and discussed in the fourth section. Finally, the conclusions are provided in the last section.

2 Background

In this section, an overview of the vehicle models, the control strategies of the path tracking task, and the related machine learning algorithms are described.

2.1 Path Tracking and Related Works

Path tracking can be categorized into three main groups: geometric, kinematic, and dynamic. Due to its simplicity, geometric path tracking is one of the most commonly used models. In a geometric vehicle model, only the dimensions and the position of the vehicle are taken into consideration with no regard to internal or external forces, velocity, or acceleration. Geometric controllers are the most common controllers in the field of path tracking due to their stability and simplicity, where the state variables are simple with the absence of the derivatives. Follow the Carrot, Pure Pursuit, and Stanley are the best-known geometric control strategies [23]. Unlike the geometric vehicle model, the kinematic model describes the motion of the vehicle taking into consideration the velocity and the acceleration with no regard to its internal forces [24], [25]. Several interesting studies have emerged in regard to kinematic controlling. Sun et al. [26] presented a study to address the problem of path tracking for the autonomous vehicle and analyze the relationship between the road model and path tracking method. De Luca et al. [27] provided a comparison study of different feedback solutions for different tasks such as path tracking and stabilization for a car-like robot (kinematic model). Kinematic and Geometric models are effective for systems where there is no need to take the internal and external forces into consideration. However, these forces should be taken into consideration under specific conditions such as a sharp trajectory curvature. Ignoring the vehicle dynamics under such conditions will negatively affect the performance and the safety aspects. In a dynamic model, the motion of

the vehicle is described with respect to its position, velocity and □ccelerateon, taking into considerations the applied internal and external forces such as the gravity force [28], [29]. Taking the effects of the vehicle dynamics into consideration naturally makes the dynamic controllers more efficient and stable than geometric and kinematic controllers [30]. However, dynamic feedback (such as the torque) is required for these control strategies, which in turns requires special types of sensors and more data processing. Consequently, dynamic controllers are more expensive in terms of the cost and computational loads [31]. An adaptive controller is also used for autonomous vehicle tasks, developed to deal with systems which have uncertain, unknown, or changeable parameters. Martins et al. [32] used an adaptive controller for a vehicle path tracking task and their proposed model used the linear and angular velocity as a reference signal. Artificial intelligence is widely used with adaptive controllers in order to improve the control decisions in terms of speed and accuracy. In paper [33], a lateral motion control method was provided where the objective of the suggested method is to maintain the yaw stability and minimize the tracking error. The control schema consists of two main modules, a steering controller to ensure the yaw stability and an artificial neural network approximator to estimate cornering stiffness uncertainty. In the field of AI in learning and control, many related works are highlighted dealing with linear and nonlinear controllers. In [34], [35], the authors of both papers use the linear controller as the classical PID and Fuzzy controller for a linear system [36], [37], while others have focused on using nonlinear controllers and learning algorithms as presented in [38], [39], [40]. Reference [41] reports a new Reinforcement Learning (RL)-based control approach that uses Policy Iteration (PI) and a metaheuristic Grey Wolf Optimizer (GWO) algorithm to train the Neural Networks (NNs). The GWO algorithm shows good results in NN training and solving complex optimization problems.

2.2 Reinforcement Learning Algorithms and Related Works

Sequential decision making problems can be formulated by Markov Decision Processes (MDPs), which is considered a bedrock of the problems that reinforcement learning solves. MDPs consist of a decision maker (agent), set of states (S), set of actions (T), transaction function (A), and reward function (R,) which can be represented as a tuple <S, A, T, R>. At each time step (t), and based on the received state ($S_t \in S$), the agent takes an action ($A_t \in A$) which represents a pair (A_t, S_t) in the next time step. Based on the taken action the environment is transitioned to a new $S_{t+1} \in S$, and the agent receives a reward $R_{t+1} \in R$, [42], [43] (see Figure 1). The cumulative reward is simply represented as a sum of the expected return at each time step. The probability of selecting an action by the agent from all possible actions at all possible states is determined by the policy (π) that the agent follows. In addition to the probability of the selection action, the value function evaluates how good it is for the agent to select an action at a given state under a policy (π), and this is called the action-value function ($q\pi$), or how good

it is for the agent to be at a given state following a policy (π), and this is called the state-value function (v_π). Equations 1 and 2 are the mathematical representations of the action-value and the state-value functions, respectively. The action-value function $q_\pi(s,a)$ is the expected reward ($\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}$) starting from state (s) at time (t), performing the action (a) and following the policy (π), where the state-value function v_π(s) is the expected reward starting from state (s) at time (t) and following the policy (π). It is worth mentioning that q_π is also referred to as the Q-function and its output is called the Q-value (the quality of taking an action). In terms of optimality, the main goal of the RL algorithm is to select the optimal policy that will yield the highest expected reward for each state. The optimal policy is associated with an optimal state-value function (v_*) and an optimal action-value function (q_*) or optimal Q-function, which are represented in equations 3 and 4, respectively. The fundamental property that the optimal Q-function (q_*) must satisfy is the Bellman equation (see equation 5), where (R_{t+1}) is the expected reward that the agent obtains by taking the action (a) at state (s), whereas $\gamma \max q_*(s',a')$ is the maximum expected discounted reward that can be received from any next state-action pair [44], [45]. Reinforcement learning is a category of machine learning that studies the behavior of an agent and focuses on how this agent might interact with its environment. The main goal of the agent is to maximize the cumulative given rewards it receives over time in order to optimize its behavior in such an environment [46]. Based on the fact that the agent is able to learn the value function estimates or/and the policies directly, RL methods can be categorized into three main methods: value-based methods, policy-based methods, and actor-critical methods [47]. All of the methods share the same strategy of determining the actions and evaluating the agent behavior, but the essential difference is where the optimality resides.

$$q_\pi(s,a) = E_\pi\left(\sum_{k=1}^{\infty}\gamma^k R_{t+k+1} \,|S_t = s,\ A_t = a\right) \tag{1}$$

$$v_\pi(s) = E_\pi\left(\sum_{k=1}^{\infty}\gamma^k R_{t+k+1} \,|S_t = s\right) \tag{2}$$

$$v_*(s) = \max_\pi v_\pi(s) \tag{3}$$

$$q_*(s,a) = \max_\pi q_\pi(s) \tag{4}$$

$$q_*(s,a) = E_\pi\left(R_{t+1} + \gamma \max_{a'} q_*(s',a')\right) \tag{5}$$

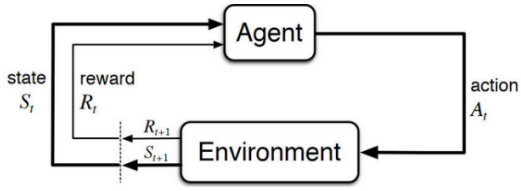

Figure 1
Markov Decision Processes

2.2.1 Value-based Algorithm

Value-based methods aim to get the optimal cumulative reward and determine the optimal policy that follows the recommendations. One of the most commonly used reinforcement learning value-based algorithms is the Q-learning method [48]. The objective of Q-learning is to find the optimal policy by learning how to find the optimal Q-value for the (s, a) pair, where the Q-values are stored in a Q-table. The Q-learning algorithm uses what is called the value iteration approach to converge the Q-function to the optimal Q-function by iteratively updating the Q-value for each (s, a) pair using the Bellman equation. With the increase in environment complexity, the state space size increases, and the performance of the Q-learning method will drop off because of the value iteration strategy that is used to update the Q-values (Q-table). The problem with large MDPs is that there are too many states and/or actions to be stored in the memory, and it is too slow to calculate the value for every individual state [49]. To overcome this problem, a function approximation is used to estimate the values instead of using the value iteration. The deep neural network is used as a function approximation and combined with the Q-learning method. This method is called Deep Q-learning, where the Deep Q-Network (DQN) approximates the Optimal-Q value [50]. The DQN model accepts the state as an input and outputs the estimated Q-value for every possible action that can be taken at that given state. After calculating the loss, the weights within the neural network are updated by stochastic gradient descent (SGD), just like in any other neural network.

2.2.2 Policy-based Algorithm

Like the value-based method, the policy-based method selects one possible action and evaluates the agent's behavior thereafter in order to achieve optimization. The essential difference between the two methods is a matter of how to achieve optimality. While the value-based method selects the optimal policy based on the optimal cumulative reward, the policy-based method directly optimizes the policy itself. The policy is parameterizes $\pi_\theta(s, a)$ and the optimization problem turns out to be finding θ, which maximizes the policy's objective function $J(\theta)$ [48]. In other words, policy-based methods learn how these parameters should change the probabilities by which different actions can be taken in different states in order to maximize the expected reward. The main advantage of policy-based methods is their effectiveness for continuous action or the high dimensional space, where the parameters of the 'parameterized policy' are adjusted instead of solving a complicated maximization in every step. The policy gradients (PG) algorithm is widely used to solve the problems of the continuous action space. The policy is represented by a parametric probability distribution (see equation 6). In the PG algorithm, the action (a) at state (s) is selected stochastically based on a vector of parameters (θ), and by adjusting these parameters, the policy is driven in the direction of increasing the cumulative reward [49]. Policy gradient is the derivatives

(vector of derivatives) of the policy's objective function $J(\theta)$ with respect to the parameters (θ) as shown in equation 7 [51]. The problem can be formalized as shown in equation 8, considering (τ) is the agent's trajectory, $R(\tau)$ is the corresponding reward, (π_θ) is the parameterized policy and $P(\tau \mid \theta)$ is the probability of the trajectory (τ) under the policy (π_θ). The policy gradients algorithm searches for the local maximum by ascending the gradient of the policy with respect to the parameters (θ). It seeks to increase the probabilities of the trajectories that give the best return, as shown in equation 9. By reformulating the probability of the trajectory $P(\tau \mid \theta)$ and decomposing the trajectory into (states – actions), the policy gradients equation can be reformulated as shown in equation 10. Instead of integrating over the spaces of both state and action as in the case of stochastic policy gradients, deterministic policy gradients (DPG) integrates only over the state space, which in turns leads to a reduced number of samples, especially in the case of applications with large action states [48]. DPG is used in the deterministic environment (no uncertainty) where it accepts a state as input and outputs a single action $\pi_\theta(s) = a$. On the other hand, the stochastic policy is always needed to explore the complete state-action space. Based on that and for sufficient exploration for the DPG algorithm, the actions are chosen according to stochastic policy behavior, while learning a deterministic target policy. The policy that the agent uses to determine its actions at a given state is called behavior policy, while the policy that the agent uses to update the Q-value is called target policy. Learning the policy can be achieved in two different algorithms, on-policy or off-policy [52]. In the case of on-policy learning, the behavior policy is the same as the target policy, while they are different in the case of the off-policy learning algorithm.

$$\pi_\theta = P[a \mid s, \theta] \tag{6}$$

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} \tag{7}$$

$$\theta^* = \arg\max_\theta J(\theta) = \max_\theta \sum\nolimits_\tau P(\tau|\theta) R(\tau) \tag{8}$$

$$\nabla_\theta J(\theta) = E_\tau \left(\nabla_\theta \log P(\tau|\theta) R(\tau)\right) \tag{9}$$

$$\nabla_\theta J(\theta) = E_\tau \left(\nabla_\theta \log \pi_\theta P(s|t)\right) \tag{10}$$

2.2.3 Actor-Critic Algorithm

Actor–critic algorithms combine the benefits of both value-based and policy-based algorithms. The essential idea is that a value function approximator (critic) is used to explicitly estimate the action-value function instead of using the return. These algorithms deal with two different sets of parameters using two different approximators, the critic and the actor. The critic updates the action-value function

parameters, while the actor updates the policy parameters based on the direction that is suggested by the critic [53]. Actor-critic algorithms use an approximate policy gradient as described in equation 11, where the $Q_W(s,a)$ is the estimated cation-value function. Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy, actor-critic reinforcement learning algorithm that searches for the optimal policy that maximizes the cumulative long-term return for the continuous action environment. DDPG uses deep neural network-based approximators [44]. In the DDPG algorithm, the actor is used to approximate the optimal policy deterministically, which is unlike the stochastic policy, where the policy learns the probability distribution rather than actions. After the action is taken by the actor, the critic evaluates that action in order to determine whether the new state is better or worse than the expectation. That can be achieved by maintaining the Q-values of the taken actions towards the target Q-values. RL has been applied to a variety of autonomous driving tasks, [54], [55], [56], [57].

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left(\nabla_\theta \log \pi_\theta(s|a)\, Q_w(s,a)\right) \tag{11}$$

2.3 Supervised Learning Compared to Reinforcement Learning

Unlike RL methods where the agent learns by interacting with the environment without any supervision data, in supervised learning, the agent learns using labeled data sets. This means that the expert is explicitly guiding the model on how to act based on the labeled data. In deep neural networks, for example, and during training, the network approximates the future outputs for the observations and then compares them with the labeled ones in order to reduce the error. Supervised learning is mainly dedicated to dealing with two main categories of tasks, classification and regression, whereas RL deals with Markov's decision processes, policy learning, and value learning. The simplicity and the speed of the convergence during the training are the advantages of supervised learning compared to reinforcement, where convergence to the optimal policy can be slow so it requires intensive time. On the other hand, the efficiency of the supervised model is greatly affected by the comprehensiveness of the training data-set. In the case of nonlinear and complex systems such as driving system tasks, sufficient training data must be ensured in order to provide an efficient and generalizable model in all complex driving environments. The use of deep learning in a variety of fields has increased recently due to new powerful processing technologies that reduce the training time and improve performance. The deep neural network algorithm is a self-optimization algorithm and it has the ability to adopt a new scenario, which enables the developers to generalize the desired models. These features make deep learning suitable for control applications within dynamic and complex environments. The computational complexity and the advantages of the learning-based methods compared to classical MPC are presented in the additional material, Section 2.3. (See: [1])

3 Design of the Controllers

This section includes the designing process of the MPC, the DNN, the DQN and the combined models. The DNN model is developed to imitate the behavior of the MPC. The deep network of the DQN model will be designed with the same structure as that of the DNN model. In the combined model, the trained DNN model is combined with a Reinforcement DDPG algorithm as a decision maker. DQN and Hybrid models were trained until the determined criteria are achieved (desired reward, number of episodes, ... etc.)

3.1 Design of the MPC Controller

Since the MPC is a model-based controller, the first step in the design process is to design the vehicle model. Figure 2 shows the global position of the vehicle, while equations 12, 13 and 14 are the mathematical representation of the vehicle dynamic. Figure 3 shows the MPC model, while the input-output signals, the parameters, and the constraints of the MPC are presented in Table 1. During the designing process, the parameters of the MPC are initiated based on standard recommendations and were tuned during the testing until a stable behaviour is achieved. A detailed explanation about MPC optimization problem, Performance specifications, control law, and parameter calculations can be found in [1], section (3.1).

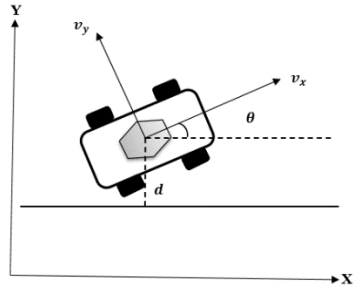

Figure 2
Global position of the vehicle

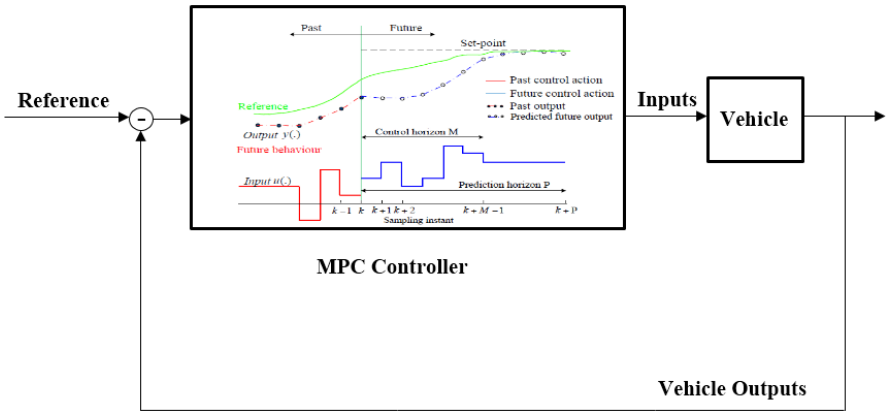

Figure 3
MPC controller design

$$\frac{d}{dt}\begin{bmatrix} v_y \\ \omega \\ d \\ \theta \end{bmatrix} = A\begin{bmatrix} v_y \\ \omega \\ d \\ \theta \end{bmatrix} + B\begin{bmatrix} \delta \\ \rho \end{bmatrix} \tag{12}$$

$$A = \begin{bmatrix} -\frac{2c_f + 2C_r}{mv_x} & -v_x - \frac{2c_f l_f - 2c_r l_r}{mv_x} & 0 & 0 \\ -\frac{2c_f l_f - 2c_r l_r}{l_z v_x} & -\frac{2C_f l_f^2 - 2C_r l_r^2}{l_z v_x} & 0 & 0 \\ 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{13}$$

$$B = \begin{bmatrix} \frac{2c_f}{m} & 0 \\ \frac{2c_f l_f}{l_z} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{14}$$

where v_x is longitudinal velocity; v_y is lateral velocity; d is lateral deviation; c_f is the corner stiffness of front tires; l_r is the distance between the rear tire and the center of the gravity; I_z is yaw moment; m is the vehicle's mass, ω is yaw rate; δ is steering angle; θ is yaw angle; c_r is the corner stiffness of rear tires; l_f is the distance between the front tire and the center of gravity; and ρ is the curvature.

Table 1
Design parameters and system constraints of the MPC controller

Internal model (vehicle)	Input signals	Steering angle (δ)
		Disturbance ($\rho\ v_x$)
	Output signals	Lateral deviation (d)
		Yaw angle (ω)
		Lateral velocity (v_y)
		Yaw rate (ω)
Parameters of MPC model	Sample time (T_s)	0.1 seconds
	Prediction horizon (P)	2 seconds
	Control horizon (M)	2 seconds
Constraints	Steering angle	[-1.04, 1.04] rad
	Changing rate	[-0.26, 0.26] rad

3.2 Design of the DNN model Using Imitation Learning

To achieve imitation learning, the DNN model was designed and structured based on the MPC model, where six observations are determined as inputs ($\theta,\ v_y, d, \omega, \rho,$ $\hat{\delta}$) and one control action (δ) was determined as an output, where ($\hat{\delta}$) is the previous control action. The detailed structure is shown in Figure 4. In regard to the training options, Adaptive Moment Estimation (ADMA) is used as an optimizer, the maximum number of Epoch is set to be 40, the mini-batch for each iteration is set to be 420, and the initial learning rate is set to be 0.01. Data preparation and training process of the MPC controller is presented in [1], Section 3.2.1

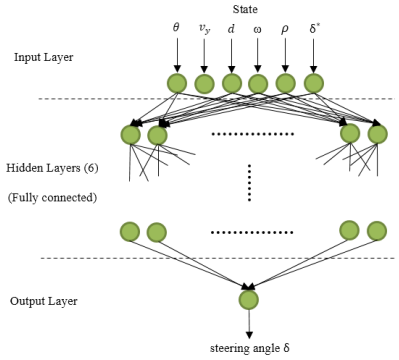

Figure 4
The DNN model structure

3.3 Design of the Reinforcement Deep Q-Learning Model

The desired DQN model is designed taking into consideration the same dynamics of the vehicle, the constraints, and the environment conditions that were used previously. The designing processes went through several steps, preparing the environment, creating and training the agent and finally testing and evaluating the performance. The environment is created using the six observations and the control action space was determined as a discrete space in the range of [-1.04, 1.04] rad, meaning that the agent can apply 121 possible actions at each state. Based on that, the deep Q-network is designed to accept the state from the environment as an input (vector with 6 observations) and outputs the estimated Q-values of each possible discrete action that can be taken at that state (vector of n=121 Q values). The detailed structure of the DQN model is shown in Figure 5. The target DQN, which is used to calculate the target Q-values is a clone of the DQN with the same structure and parameterization. The training details of the DQN model are presented in [1] Section 3.3.1.

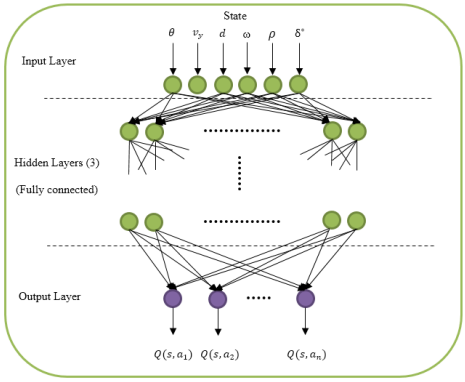

Figure 5
The reinforcement DQN structure

3.4 Design of the Combined (Supervised–RL) Model

The same vehicle's dynamics, constraints, environment conditions and state space are used to design and test the combined model. The continuous actions space is determined to be in the range of [-1.04, 1.04] rad. In order to create the agent, beside having the trained DNN model as an actor, the critic is created based on the actions-observations specifications, where the neural network is structured to accept two inputs (state-action) and one output (the corresponding expected long-term reward $Q\ (s, a \mid \theta^Q)$, and 3 hidden layers. Figure 6 shows the detailed structure of the combined model. The training process is presented in Section 3.4.1 of [1]

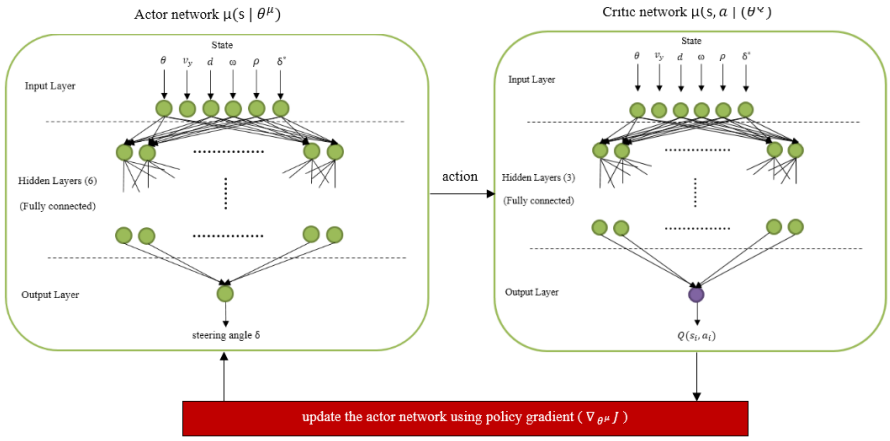

Figure 6
The structure of the actor-critic networks - combined model

4 Results and Discussion

The implementations of the designed models were performed using the same vehicle model and subjected to the same constraints, environmental conditions, and initial state. The performance is analyzed and evaluated taking into consideration the performance of the MPC controller as a reference behavior. The efficiency is discussed based on different indicators: the ability of the controllers to drive the vehicle along the desired trajectory in the first place, the time needed to reach a stable state, and the smoothness of the driving system. The obtained results in Figure 7 clearly show that the trained DNN and the MPC controller behave similarly with very small output deviation, where the maximum difference is approximately 0.0094 rad (0.53 degrees). The behavior is evaluated based on the response of the vehicle to the controllers' outputs. Figure 8 shows vehicle response to the control actions of the MPC and the DNN models in terms of lateral deviation and Figure 9 shows that both controllers (MPC-DNN) were able to follow the desired trajectory by driving the lateral deviation and yaw angle to be very close to zero. Additionally, and taking into consideration the control system characteristics, the results clearly show that both controllers were able to reach the stable state at almost the same time with the same amount of overshooting. These results prove that the trained DNN model was able to imitate the behavior of the traditional MPC controller successfully. Based on that, the performances of the reinforcement DQN model and combined model are compared to the DNN model in order to evaluate the best result achieved by the machine-learning-based models. Figure 10 shows that the three models responded differently to the same initial state. Despite these differences, Figures 11 and 12 show that the reinforcement DQN and the combined models were able to track the desired trajectory with different control system characteristics

(steady state time and overshooting). The detailed results showed that the combined model responded in a way that improved the smoothness of the driving system by reducing the overshooting (with hardly any overshooting in the case of lateral deviation) and drove the lateral deviation to be very close to zero (0.003 m) in a reasonable time, compared to the DNN model which achieved 0.0009 m as a final value of the lateral deviation at almost the same time but with higher overshooting and thus higher lateral deviations. The DQN model was not as efficient as the other models; its behavior led to higher overshooting and drove the lateral deviation to a final value of 0.01 m. As a result, and taking all the performance indicators into considerations, one can state that the combined model provided the best result and achieved the expected optimization by demonstrating accurate control actions (steering angles) that steer the vehicle along the desired trajectory efficiently in a reasonable time and improve the robustness of the driving system, while the DQN model, which is completely based on an RL algorithm, was not as efficient as the other two models (the supervised DNN or the combined model). The promising results that are provided by the reinforcement learning methods (DQN and Hybrid model) emphasize the importance of devoting more efforts to transferring them into practice as an efficient alternative to classical control methods.

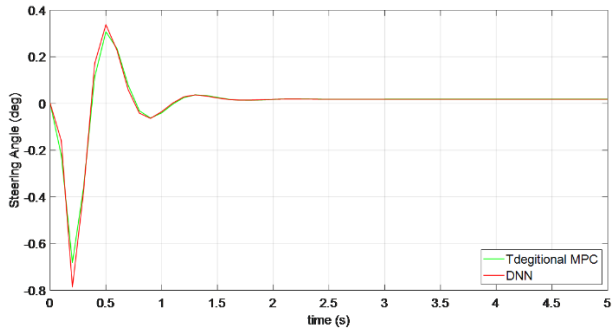

Figure 7
Comparison of the estimated steering angles of the MPC and the DNN models

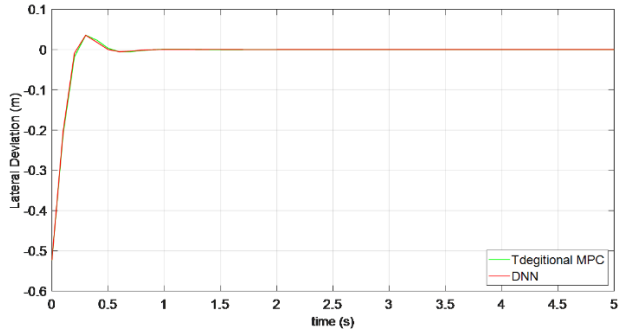

Figure 8
Vehicle response to the control actions of the MPC and the DNN models - lateral deviation

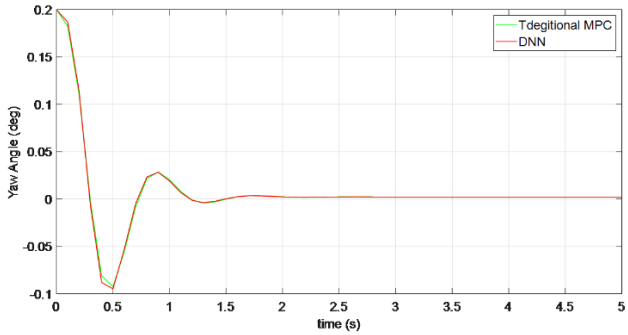

Figure 9
Vehicle response to the control actions of the MPC and the DNN models - yaw angle

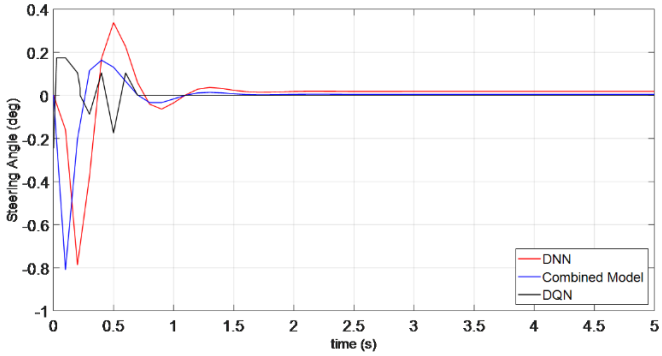

Figure 10
Comparison of the estimated steering angles of the DNN, DQN, and combined models

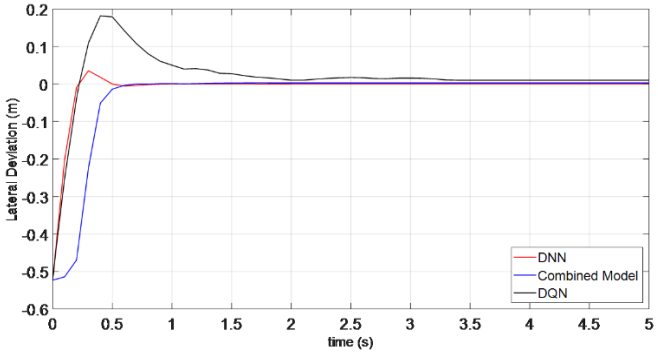

Figure 11
Vehicle response to the control actions of the DNN, DQN - lateral deviation

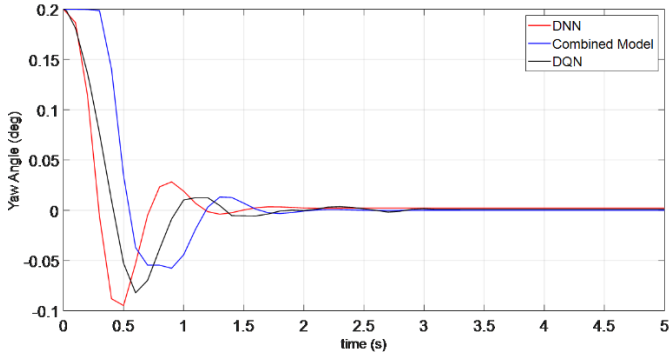

Figure 12
Vehicle response to the control actions of the DNN, DQN - yaw angle

Conclusions

In this work three different machine learning-based models were designed to perform an automated path-tracking task: a DNN model to imitate the behavior of the traditional MPC controller, a reinforcement learning DQN model, and a hybrid model. The hybrid model was designed to optimize the performance by combining the trained DNN model with the reinforcement learning model, where the DNN network was used as a decision-maker along with the critic network that evaluates the actions taken. The results showed that all three models were able to drive the vehicle along the desired path. The combined model was able to provide the desired optimization by driving the vehicle to the reference speed more smoothly and within a reasonable time. This work shows the efficiency of combining supervised and reinforcement learning to leverage the advantages of both algorithms, where the supervised learning speeds up the learning process and the reinforcement learning improves self-adaptation to new states that the model was not faced within the training process, which increases efficiency in the complex driving environment.

References

[1] A. Reda, R. Benotsmane, A. Bouzid, J. Vásárhelyi: Additional material for a Hybrid Machine Learning-Based Control Strategy for Autonomous Driving Optimization paper Acta1034; submitted to Acta Polytechnica Hungarica ISSN, https://drive.google.com/drive/folders/14hjWFoAxhHnWi18W-iO23QZe2btwF-de, See: Additional-material-ACTA-1034.pdf, 2023, p. 11

[2] C. Thorpe, M. Herbert, T. Kanade, S. Shafter: Toward autonomous driving: the CMU Navlab. II. Architecture and systems, in IEEE Expert, Vol. 6, No. 4, 1991, pp. 44-52

[3] E. Dickmanns, A. Zapp: Autonomous high speed road vehicle guidance by computer vision 1, IFAC Proc. Volumes, Vol. 20, No. 5, 1987, pp. 221-226

[4] SAE: On-Road Automated Vehicle Standards Committee, Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems, Technical Report, 2014, pp. 1-6

[5] U.S Department of Transportation: Preparing for the Future of Transportation: Automated Vehicles 3.0, 2018, https://www.transportation.gov/av/3, last visited December 2022

[6] B. Aelbachir, J. C. Smal, J. C. Blosseville, D. Gruyer: Simulation-driven validation of advanced driving-assistance systems, Procedia-Social and Behavioral Sciences, Vol. 48, 2012, pp. 1205-1214

[7] S. Hima, S. Glaser, A. Chaibet, B. Vanholme: Controller design for trajectory tracking of autonomous passenger vehicles, 4th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 1459-1464

[8] X. Li, Z. Wang, J. Zhu, Q. Chen: Adaptive tracking control for wheeled mobile robots with unknown skidding, 2015 IEEE Conference on Control Applications (CCA), 2015, pp. 1674-1679

[9] X. Wu, P. Lou, D. Tang, J. Yu: An intelligent-optimal predictive controller for path tracking of Vision-based Automated Guided Vehicle, 2008 International Conference on Information and Automation, 2008, pp. 844-849

[10] C. Sun, X. Zhang, Q. Zhou, Y. Tian: A Model Predictive Controller With Switched Tracking Error for Autonomous Vehicle Path Tracking, in IEEE Access, Vol. 7, 2019, pp. 53103-53114

[11] J. B. Rawlings, D. Q. Mayne: Model Predictive Control: Theory and Design, Nob Hill Publishing, Madison, 2009

[12] M. Brown, J. Funke, S. Erlien, J. C. Gerdes: Safe driving envelopes for path tracking in autonomous vehicles, Control Engineering Practice, vol. 61, 2017, pp. 307-316

[13] A. Reda, A. Bouzid, J. Vásárhelyi: Model predictive control for automated vehicle steering, Acta Polytechnica Hungarica, Vol. 17, 2020, pp. 163-182

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton: ImageNet classification with deep convolutional neural networks, Commun. ACM, Vol. 60, No. 6, 2017, pp. 84-90

[15] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, A. Acero, Recent advances in deep learning for speech research at Microsoft, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8604-8608

[16] S. Kuutti, R. Bowden, Y. Jin, P. Barber, S. Fallah: A Survey of Deep Learning Applications to Autonomous Vehicle Control, in IEEE Transactions on Intelligent Transportation Systems, Vol. 22, No. 2, 2021, pp. 712-733

[17] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, C. Cadena: From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots, In 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 1527-1533

[18] R. Girshick, J. Donahue, T. Darrell, J. Malik: Rich feature hierarchies for accurate object detection and semantic segmentation, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587

[19] M. Batta: Machine learning algorithms - a review. International Journal of Science and Research (IJSR), 2020, pp. 381-386

[20] Y. Li, H. Li, Z. Li, H. Fang, A. Sanyal, Y. Wang, Q. Qiu: Fast and Accurate Trajectory Tracking for Unmanned Aerial Vehicles based on Deep Reinforcement Learning, 2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2019, pp. 1-9

[21] S. Gu, E. Holly, T. Lillicrap, S. Levine: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, IEEE International Conference on Robotics and Automation, 2017, pp. 3389-3396

[22] A. B. Martinsen, A. M. Lekkas: Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models, OCEANS 2018 MTS/IEEE Charleston, 2018, pp. 1-8

[23] S. Bacha, M. Y. Ayad, R. Saadi, A. Aboubou, M. Bahri, M. Becherif: Modeling and control technics for autonomous electric and hybrid vehicles path following, 2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B), 2017, pp. 1-12

[24] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, L. B. Becker: A Predictive Controller for Autonomous Vehicle Path Tracking, IEEE Transactions on Intelligent Transportation Systems, Vol. 10, No. 1, 2009, pp. 92-102

[25] J. Wang, J. Steiber, B. Surampudi: Autonomous ground vehicle control system for high-speed and safe operation, International Journal of Vehicle Autonomous Systems, Vol. 7, No. 1, 2009, pp. 18-35

[26] Z. Sun, Q. Chen, Y. Nie, D. Liu, H. He: Ribbon Model based path tracking method for autonomous land vehicle, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 1220-1226

[27] A. De Luca, G. Oriolo, C. Samson: Feedback control of a nonholonomic car-like robot, In Robot motion planning and control, Springer, Berlin, 1998, pp. 171-253

[28] N. H. Amer, H. Zamzuri, K. Hudha, Z. A. Kadir: Modelling and control strategies in path tracking control for autonomous ground vehicles: A review

of state of the art and challenges, Journal of Intelligent $\&$ Robotic Systems, Vol. 86, No. 2, 2017, pp. 225-254

[29] E. Lucet, R. Lenain, C. Grand: Dynamic path tracking control of a vehicle on slippery terrain, Control Engineering Practice, Vol. 42, 2015, pp. 60-73

[30] C. Poussot-Vassal, O. Sename, L. Dugard, S. M. Savaresi: Vehicle dynamic stability improvements through gain-scheduled steering and braking control, Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility, Vol. 49, No. 10, 2011, pp. 597-1621

[31] P. Zhao, J. Chen, T. Mei, H. Liang: Dynamic motion planning for autonomous vehicle in unknown environments, IEEE Intelligent Vehicles Symposium (IV), 2011, pp. 284-289

[32] F. N. Martins, W. C. Celeste, R. Carelli, M. S.-Filho, T. F. B.-Filho: An adaptive dynamic controller for autonomous mobile robot trajectory tracking, Control Engineering Practice, Vol. 16, No. 11, 2008, pp. 1354-1363

[33] X. Ji, X. He, C. Lv, Y. Liu, J. Wu: Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits, Control Engineering Practice, Vol. 76, 2018, pp. 41-53

[34] C. Sun, X. Zhang, L. Xi, Y. Tian: Design of a path-tracking steering controller for autonomous vehicles, Energies, Vol. 11, No. 6, 2018, p. 1451

[35] A. J. Babqi and B. Alamri: A comprehensive comparison between finite control set model predictive control and classical proportional-integral control for grid-tied power electronics devices, Acta Polytechnica Hungarica, Vol. 18, No. 7, 2021, pp. 67-87

[36] R. E. Precup, S. Preitl, E. M. Petriu, J. K. Tar, M. L. Tomescu, C. Pozna: Generic two-degree-of-freedom linear and fuzzy controllers for integral processes, J Franklin Inst, Vol. 346, No. 10, 2009, pp. 980-1003

[37] Zs. Preitl, R. E. Precup, J. K. Tar, M. Takács: Use of Multi-parametric Quadratic Programming in Fuzzy Control Systems, Acta Polytechnica Hungarica, Vol. 3, No. 3, ISSN 1785-8860, 2006, pp. 29-43

[38] T. Chen, A. Babanin, A. Muhammad, B. Chapron, and C. Chen: Modified Evolved Bat Algorithm of Fuzzy Optimal Control for Complex Nonlinear Systems, ROMJIST, Vol. 23, No. 672, 2020, pp. 28-40

[39] H. Redjimi, J. K. Tar: Multiple Components Fixed Point Iteration in the Adaptive Control of Single Variable 2nd Order Systems, Acta Polytechnica Hungarica, Vol. 18, No. 9, 2021, pp. 69-86

[40] A. Reda, J. Vásárhelyi: Model-Based Control Strategy for Autonomous Vehicle Path Tracking Task, Acta Universitatis Sapientiae, Electrical and Mechanical Engineering, Vol. 12, No. 1, 2020, pp. 35-45

[41] A. Zamfirache, R. E. Precup, R. C. Roman, and E. M. Petriu: Policy Iteration Reinforcement Learning-based control using a Grey Wolf Optimizer algorithm, Inf Sci, Vol. 585, 2022, pp. 162-175

[42] S. Sharifzadeh, I. Chiotellis, R. Triebel, D. Cremers: Learning to drive using inverse reinforcement learning and deep Q-networks, arXiv preprint arXiv:612.03653, 2016, Available: http://arxiv.org/abs/1612.03653

[43] M. L. Puterman: Markov decision processes: discrete stochastic dynamic programming, John Wiley $\&$ Sons, New York, 1994

[44] R. A. Howard: Dynamic Programming and Markov Processes, Cambridge, MA, USA: MIT Press, 1960

[45] R. S. Sutton, A. G. Barto: Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998

[46] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau: An Introduction to Deep Reinforcement Learning, Foundations and Trends in Machine Learning, Vol. 11, No. 3-4, 2018, pp. 219-354

[47] Y. Li: Deep reinforcement learning: An overview, arXiv preprint arXiv:1701.07274, 2017

[48] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep Reinforcement Learning: A Brief Survey, in IEEE Signal Processing Magazine, Vol. 34, No. 6, 2017, pp. 26-38

[49] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra: Continuous control with deep reinforcement learning, ICRL, 2016

[50] V. Mnih et al.: Human-level control through deep reinforcement learning, Nature, Vol. 518, No. 7540, 2015, pp. 529-533

[51] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour: Policy gradient methods for reinforcement learning with function approximation, Advances in Neural Information Processing Systems, 2000, pp. 1057-1063

[52] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller: Deterministic policy gradient algorithms, In International conference on machine learning, 2014, pp. 387-395

[53] J. C. Jesus, J. A. Bottega, M. A. S. L. Cuadros, D. F. T. Gamarra: Deep Deterministic Policy Gradient for Navigation of Mobile Robots in Simulated Environments, International Conference on Advanced Robotics (ICAR), 2019, pp. 362-367

[54] A. E. Sallab, M. Abdou, E. Perot, S. Yogamani: End-to-end deep reinforcement learning for lane keeping assist, MLITS, NIPS Workshop, Vol. 2, 2016

[55] P. Wang, C.-Y. Chan, A. de La Fortelle: A reinforcement learning based approach for automated lane change maneuvers, IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 1379-1384

[56] Z. Huang, X. Xu, H. He, J. Tan, Z. Sun: Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 49, No. 4, 2017, pp. 730-741

[57] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, J. W. Choi: Autonomous braking system via deep reinforcement learning, IEEE 20th International conference on intelligent transportation systems (ITSC), 2017, pp. 1-6