Hindawi
Modelling and Simulation in Engineering
Volume 2021, Article ID 4641450, 10 pages
https://doi.org/10.1155/2021/4641450

Research Article

Actor-Critic Traction Control Based on Reinforcement Learning with Open-Loop Training

M. Funk Drechsler,[1] **T. A. Fiorentin**,[2] **and H. Göllinger**[1]

[1]*Technische Hochschule Ingolstadt, Ingolstadt, Germany*
[2]*Centro Tecnológico de Joinville, Universidade Federal de Santa Catarina, Joinville, Brazil*

Correspondence should be addressed to M. Funk Drechsler; maikoldrechsler@gmail.com

Received 24 July 2021; Accepted 25 November 2021; Published 7 December 2021

Academic Editor: Fahad Al Basir

The use of actor-critic algorithms can improve the controllers currently implemented in automotive applications. This method combines reinforcement learning (RL) and neural networks to achieve the possibility of controlling nonlinear systems with real-time capabilities. Actor-critic algorithms were already applied with success in different controllers including autonomous driving, antilock braking system (ABS), and electronic stability control (ESC). However, in the current researches, virtual environments are implemented for the training process instead of using real plants to obtain the datasets. This limitation is given by trial and error methods implemented for the training process, which generates considerable risks in case the controller directly acts on the real plant. In this way, the present research proposes and evaluates an open-loop training process, which permits the data acquisition without the control interaction and an open-loop training of the neural networks. The performance of the trained controllers is evaluated by a design of experiments (DOE) to understand how it is affected by the generated dataset. The results present a successful application of open-loop training architecture. The controller can maintain the slip ratio under adequate levels during maneuvers on different floors, including grounds that are not applied during the training process. The actor neural network is also able to identify the different floors and change the acceleration profile according to the characteristics of each ground.

1. Introduction

A preliminary study of the actor-critic reinforcement learning algorithm to control the traction of an electrical vehicle had been presented at the conference XXII ENMC—Encontro Nacional de Modelagem Computacional at Juiz de Fora in Minas Gerais, Brazil [1]. In the previous research, a closed-loop approach had been used to train the neural network, while in the current paper, an open-loop method is proposed to permit a future dataset collection and the controller implementation on the real vehicle.

Nowadays, the car market is very competitive. Consumers are increasingly looking for efficient, comfortable, and safe vehicles [2–4]. To assist the driver, many safety functions have been developed, namely, autonomous emergency braking (AEB), electronic stability control (ESC), adaptive cruise control (ACC), and many others. Generally, the problems that the driver assistance systems need to control are complex and difficult to model. The optimization methods or reinforcement learning algorithms are options to solve these complex or unmodeled problems [5, 6].

The reinforcement learning (RL) algorithms allow the prediction of future states based on the past environment interaction and evaluation of the chosen actions [7]. The generic solution of an RL algorithm consists of the construction of a policy π that maps the correlations between the states and the action to be taken. In some cases, this policy is a very simple equation, while in some cases, can be a complex search process [8, 9].

Inside the RL structure, the reward function also exerts an important role correlating the state or state action to positive or negative feedbacks named reward and punishment,

respectively. Based on the feedbacks, policy chooses the actions which maximize the rewards or minimize the punishment [10].

While the reward function evaluates a state in an immediate sense, the value function evaluates it in the long run. The value function represents the amount of reward that the agent can expect in the future starting from the current state. This function prevents the agent from choosing an action that immediately gives a great reward but is preceded by low rewards [8].

Using the collected data to comprehend the actions which return the best value function, the RL algorithm can improve its control abilities during the time. One of the promising uses of the RL framework was Samuel's checkers, a virtual checkers game where the automatic player learns how to play just based on the game rules, a sense of direction, and parameters that show the targets of the game [11].

To create a general RL algorithm that does not need a model of the system, Watkins developed the Q-learning method in 1989 [12]. The Q-learning converges when evaluating a discrete case, with a finite number of actions and states. The Q-learning algorithm was applied with success by Harmon et al. [13] in a differential game. The game evaluated a strategy of airplane and missile simulation, in which the airplane avoids the missile, and the missile pursues the airplane. Applying the Q-learning combined with residual-gradient technics, the authors achieve excellent results, being able to achieve low levels of reward in a specific time step to significantly increase the reward during the next steps.

Recently, the combination of neural networks with these algorithms permits the evaluation of continuous states, showing excellent performance on game playing [14] and systems control [15]. However, the real-time implementation of these algorithms is limited due to the iterative effort necessary to find the action which corresponds to the maximum value function. One example of this limitation is the research of de Amaral et al. [16] which implemented an electronic stability control (ESC) strategy based on an RL algorithm. The author uses the IPG CarMaker virtual environment to generate the dataset and testing; however, the implemented algorithm, which finds the higher reward and consequently the better action, presents runtime limitations.

To handle more complex activities, the actor-critic algorithms include in addition to the approximation of value function, an approximation of the correlation state action [17]. The first advantage of the actor-critic algorithm is the possibility to work with many actions since there is no need to consider all actions to select one of them. The second advantage is the possibility to learn stochastic policies from scratch, enabling the application of the same algorithm for controlling different physical systems. Due to this, the actor-critic algorithm shows successful applications in the control area as robot movement [18].

Based on the success of these algorithms, recent research in the automotive area focused on its implementation, highlighting the research from Zhao et al. [19]. The authors trained an actor-critic network named model-free optimal control (MFOC) to work as adaptative cruise control, which can control the acceleration and brake of the vehicle. The MFOC is based on Q-learning rules with two networks to critic and actor parts, which were trained alternatively until convergence. The converged actor network was used to control the vehicle in a hardware in the loop simulation. The results present a better control response than the widely used PID controller [19].

Due to the risk of application of these algorithms in real plants, simulated environments have been implemented to evaluate the vehicle response to autonomous driving functions based on RL control [20, 21]. In this area, it is possible to highlight the research from El Sallab et al. [22] which applied the simulated environment Torcs as an exploration environment. In this case, an actor-critic controller with two networks was applied to control the lateral behavior of the vehicle. The results show that the actor-critic can smoothly control the vehicle steering [22, 23].

In the active safety area, the test of an antilock brake system (ABS) in real benches showed adequate results with an actor-critic based control [7]. An actor-critic controller also presented adequate behavior during the traction control of an electric vehicle in a simulated environment. In this research, the actor-critic algorithm was able to avoid the inadequate slip-on different floors with a single control network [1]. The actor-critic RL implementation also overcame the challenges of traction control development as the nonlinearities of tires and the simplicity necessary to real-time applications [1].

Despite the good results in the simulated environments and the large application of the actor-critic approach, some authors emphasize the challenge of implementing RL algorithms directly on the real plant [24]. The full control of the vehicle commands for example can generate unsafe behaviors as accelerating the vehicle against obstacles, or driving the vehicle to its dynamic limits, causing accidents. A possible option to overcome this difficulty is initially learning the policy in simulation and subsequently fine-tuning the policy in the final system [9]. The fine-tuning of the controller is required due to the simplification and limited representation of the plant models in the simulation, which cause different behaviors of the controller when controlling the virtual model or the real vehicle. However, as appointed by Funk Drechsler et al. [1], even the fine-tuning can be challenging due to iterative effort, limited processing power inside the vehicle, and the possible generation of unsafe maneuvers as cited before.

The present research aims to investigate the application of an open-loop actor-critic RL algorithm to control the traction of a rear-wheel-driven electric vehicle. The traction control exerts an important function of avoiding the wheel slipping during acceleration, improving the driving behavior. The evaluation considers the longitudinal behavior of the vehicle, including a nonlinear description of the tires, and permits understanding the applicability of the algorithm to nonlinear systems control. In the proposed open-loop training process, the data are collected from the vehicle without the influence of the controller, and the missing data are substituted by a random input. It avoids unsafe behaviors of the vehicle during the iterative training process. Once the

data are directly collected from the real vehicle, no fine-tuning is necessary, and the controller can be verified before being embedded in the vehicle.

The next section describes the architecture of the networks, the training process, and the methodology applied in the design of the experiments (DOE) to understand the effect of the data used during the training process. After that, the results obtained from the DOE are presented and the performance of the controllers are evaluated on different floor conditions. Finally, the conclusions and the outlooks of the research are explained.

2. Methodology

The methodology consists of training a neural network-based antislip control using data collected on two grounds with different frictions. Furthermore, the performance of the RL controller on different floors is evaluated. The simulation and training occur in a MATLAB/Simulink environment which considers the dynamic of electrical motor, ground friction, and the longitudinal behavior of the vehicle. Despite using simulated data, the nonlinearities of the tires were described by a look-up table based on the experimental curve from Braess et al. [25 , p. 410], which correlates the slip ratio and the friction of the ground including the nonlinearities of an exemplary tire.

The implemented critic is composed of a network with two hidden layers and twenty nodes in each layer, while the actor has the same number of hidden layers and just twelve hidden nodes in each layer. The network architecture including input and output data is the same applied by Funk Drechsler et al. [1]. The critic and actor networks use the Tangent-Sigmoid activation function in the hidden layers and linear function in the output layer.

The input states for actor and critic networks include a vector with vehicle velocity, motor current, vehicle acceleration, accelerator pedal position, and wheel velocity. The states are normalized considering the maximum and minimum possible values of each variable as limits. Due to the difficulty of measuring the real vehicle velocity, the possibility of removing this parameter as a controller input is also analyzed. The output action of the controller works as a new accelerator pedal position, which is sent to the electric motor controller.

The physical quantities selected as input enables the controller to identify the wheel slip by the difference of the velocities, the torque of the motor by the current, and the amount of torque being transferred to the ground by the acceleration of the vehicle. Furthermore, the acceleration or deceleration desired by the driver is given by the accelerator pedal position. In this way, the markovian decision process can be applied in this scenario, in which given the current slip of the wheel, acceleration of the vehicle, and torque in the motor, a given acceleration of the vehicle results in the same next state. Even if different floors frictions are evaluated the correlation of velocity, motor torque and acceleration should be enough to distinguish the ground conditions.

The creation of the neural controller is given by an open-loop training process, in which data collected from the vehicle without the iteration of the neural controller is used to train the controller. The generation and treatment of data and finally the training of the network are described in the next subsections.

2.1. Value Function Suppression. The main reason for applying iterative training methods is the exploration necessary for the improvement of the value function. In this case, the rewards calculated in the last iteration are used to iteratively update the value function as present in Equation (1) [26]. This enables the value function to consider all future rewards in addition to the current one. In this way, the function can choose actions that return a poor reward in a short period to reach high rewards in long-term.

$$Q_{k+1}(s,a) = r(s,a) + \gamma\, Q_k\left(s', \mu_k\left(s'\right)\right). \tag{1}$$

In Equation (1), the value function, also called Q-value, Q,is updated during the k^{th} iteration as a function of states s and actions a. The update is given by the reward r plus the old value function evaluated at the next state s' and the next action. The action is directly obtained by the policy μ evaluated at the next state. The discount factor γ multiplies the expected next Q-value to create a smooth update of the function and to permit the operation in endless horizons.

To permit the open-loop training applications, a myopic value function can be implemented. In this approach, the discount factor γ is null, and the value function or Q-value is equal to the reward as presented in Equation (2).

$$\mathrm{Q_{k+1}(s,a) = r(s,a)}. \tag{2}$$

This simplification suppresses the evaluation of the expected future rewards. Due to that, the value function responsible for criticizing the performance of the policy μ will not force the police to choose immediate worse rewards to achieve better rewards in the future. This approach can be unapplicable in specific implementations as playing chess since the actor does not avoid future punishments to receive an immediate good reward. The advantage of this learning process, on the other hand, is the possibility of reducing the exploration for training the value function and the actor, given that future rewards do not influence the current decision.

For the proposed controller, which avoids the inadequate sleep of the wheels, the predicted future rewards do not have a significant influence on the current action. Furthermore, the evaluations of the discount factor researched by Funk Drechsler et al. [1] indicate better behavior of myopic training processes for this specific implementation.

2.2. Data Generation and Processing. In closed-loop training, the neural controller is initialized with random weights, and it explores the environment in a trial and error fashion. Using this implementation in a real vehicle is dangerous, seeing that the vehicle can accelerate in an unexpected way

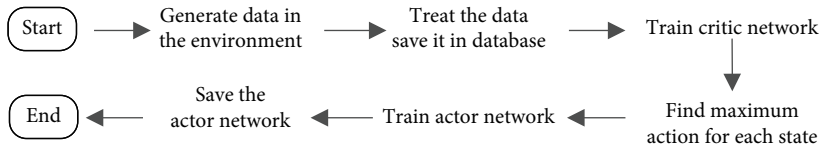

Figure 1: Open-loop training process.

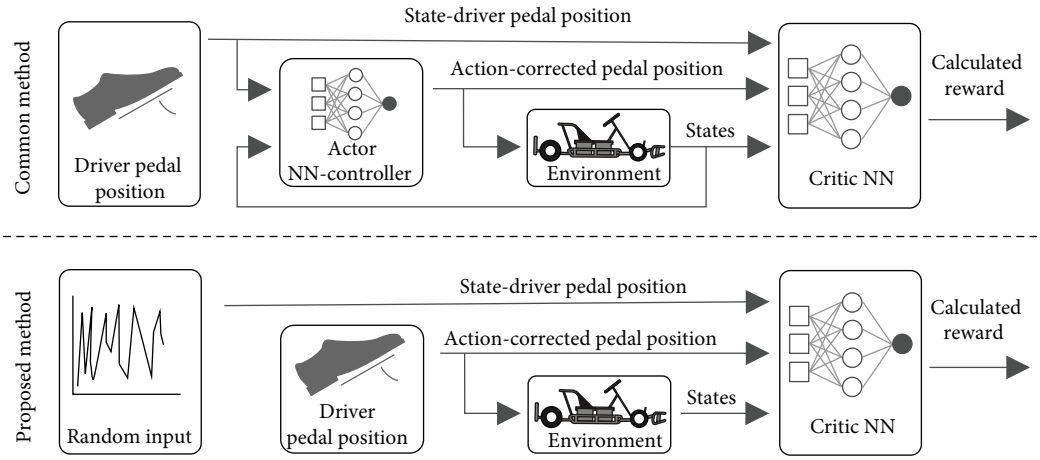

Figure 2: Correlation between environment and input data.

causing accidents. Furthermore, an iterative implementation is necessary to train the network, in which powerful onboard computers or huge effort on the cyclic collection and training of the data is required. Due to the myopic implementation, the iterative training process can be simplified, resulting in the straight training process shown in Figure 1. It enables that the data are collected by a professional driver without any risk for the vehicle, driver, test facilities, or equipment.

As presented in Figure 1 instead of having an iterative process as shown by Funk Drechsler et al. [1], the data is collected just once. The data are then treated and implemented in the training process. Once the actor network is trained, it is ready to be tested and implemented in the vehicle. This simplification of the training process supports easier data collection. Figure 2 shows a comparison between the common method and the proposed approach.

In the common method shown in Figure 2, the accelerator pedal position from the driver is fed to the actor NN, which is initialized with randomized weights and based on the current state sends the action to the environment (vehicle). The vehicle interacts with the action given, reaching a new state. Based on the action and state change, the reward and value function are calculated. This data is saved for future training or in the case of online training, and the neural networks can be directly updated.

In the proposed method represented in Figure 2, the driver has total control of the vehicle and chooses the pedal position which will control the vehicle. This action is saved as the output of the controller together with the generated states and rewards. This procedure considers that just the action (corrected pedal position) correlates with the environment and next states. In this way, the accelerator pedal signal needs to have a smoother variation, which generates different ranges of velocities, accelerations, and motor torque. On other hand, the accelerator position chosen by the driver has only a mathematical correlation in the network; thus, a completely random variation does not influence other states. Despite that a simulation is being used to prove the concept, this approach can be easily applied in a real environment, in which the accelerator pedal can be directly implemented as environment input (action), avoiding the necessity of iterative training and possible complications by unexpected behaviors of the controller.

To generate the proposed data in the simulated environment, a virtual drivel pedal position is created by a combination of sinus and step in different phases with a small noise to imitate a random driver behavior. It permits the variation of possible states tested during the training process. The data is generated in simulation with a frequency of 50 Hz and saved as a Matlab file for the training process. Data is generated representing a specific period of maneuver time; after this, all collected data is normalized to reduce the difference between the physical units, improving the network training as proposed by Ioffe and Szegedy [27]. In the sequence, the reward is calculated based on Equation (3), where GP is the accelerator pedal position selected by the driver, slip is the slip ratio, v_x is the vehicle velocity, and v_r is the wheel velocity.

$$r(s,a) = \begin{cases} 0 \text{ if } 0 > \text{GP} > 1, \\ 0, \text{ if GP} + 0.05 < a, \\ 0, \text{ if } \left|slip'\right| > 0.2, \\ 1 - |a - GP|, \text{ if } \left|\text{slip}'\right| \leq 0.2 \text{ or } \left(v_x' \leq 0.05 \text{ and } v_r' \leq 0.0625\right). \end{cases} \tag{3}$$

Table 1: Training stop criteria.

Maximum number of epochs	1000
Minimum gradient	1e-7
Maximum ε	1e -10
Maximum validation failure	6

Table 2: Factors of the DOE for the proposed algorithm.

Factors	Low level	High level
Step	No	Yes
Reward	-1	0
Amount of data	25 min	50 min

As presented in Equation (3), the reward function evaluates if the action is inside the nominated range and if it is lower than the acceleration desired by the driver. In case these statements are accomplished, the slip ratio of the next state is evaluated, when it is less than 0.2, the reward is calculated by the difference of the desired acceleration and the output of the controller; in all other cases, the reward is null or minus one according to the data evaluation explained in details in further sections.

2.3. Neural Network Training. With the collected states, actions, and calculated rewards, the critic network can be trained to define the reward of every combination of states and actions. The Levenberg-Marquardt optimization method is applied to minimize the normalized mean square error in the neural network. This method consists of a combination of simplified Newton and gradient descendent methods [28].

The criteria used to stop the training is given in Table 1, in which an epoch corresponds to passing the whole dataset through the network and the validation failure consists of the number of consecutive times that the validation performance degrades. Furthermore, the scalar ε permits to change between the Newton and gradient descendent methods. If ε is null, the optimization behaves as a Newton descendent algorithm, when ε is large the behavior is similar to a simple gradient descendent method. In this way, ε decreases at each step that the algorithm presents a reduction of the error and increases when the error increase. Due to that, the scalar ε increases with the increment of the error indicating that the training is diverging.

To find the correct correlation between states and actions, each state of the minibatch needs a correspondent best action. In this process, each one of the states has the action interactively varied inside the whole range until finding the best reward and the correspondent best action. Appling the states and the correspondent best actions, the actor network is trained using the same process applied to the critic training.

2.4. Data Influence. In closed-loop approaches, the exploration is done by the actor itself. Thus, the actor checks all necessary states and actions to improve the controller quality. However, in the proposed method, the data is generated by a driver beforehand, and this approach also needs to explore as many states as possible. To obtain these results, the input generated in the simulation includes the sum of noise, sinusoidal, and step functions that permits varying the vehicle velocity, acceleration, and motor torque.

To understand how the data used to train the neural network affects the final performance of the controller, a factorial design of experiments (DOE) is applied. The implemented DOE is a two levels planning with k factors and 95% of confidence interval. The DOE runs in the Minitab 18, evaluating the effects on the training time and the mean reward. The performance of the trained network is evaluated in a 20 seconds maneuver simulating the vehicle driver on two different floors with a combination of step and sinus as throttle pedal input. The factorial DOE uses 3 factors, and a full-resolution test is applied without replications nor center point. Table 2 presents the factors and levels utilized during the research.

As presented in Table 2, all evaluated factors are related to data generation, aiming to improve the reliability and quality of the data. Among the factors, the step consists of the application of a unitary step with a period of 20 seconds and a pulse width of 50%. This signal multiplies the accelerator data before the noise application permitting to keep the pedal position around zero, reducing the vehicle velocity. This strategy permits the creation of data that simulates low vehicle velocity states. Figure 3 presents/shows the accelerator pedal signal without (a) and with step factor(b).

The reward factor in Table 1 is related to the punishment given in case bad behavior is computed. In the high level of this factor, Equation (3) is kept the same, while in the low level, all the 0 rewards of Equation (3) are changed to punishments -1. This factor enables the evaluation of the effect of large and small differences between good and bad rewards on the performance of the controller.

Finally, the factor amount of data is respective to the amount of time at which data is collected, aiming to recognize the necessary amount of data for training the network. This factor has similar importance as the exploration in closed-loop training, given that a bigger amount of data should increase the presence of different conditions. Once the simulation runs at 50 Hz, the 25 and 50 minutes of driving correspond to 75000 and 150000 transitions, respectively.

The controllers with and without the velocity as input are evaluated separately, permitting to obtain optimized controllers in both cases. Finally, the controllers with the best parameters are evaluated on ice, snow, dry, and wet asphalt. The final comparison evaluates the slip ratio of the tires with the maneuver including a combination of sinus and step signal as shown in Figure 4. The next section presents the results obtained from the application of the described methodology.

3. Results

The results are divided into two subsections which include the evaluation of the data influence on the performance of

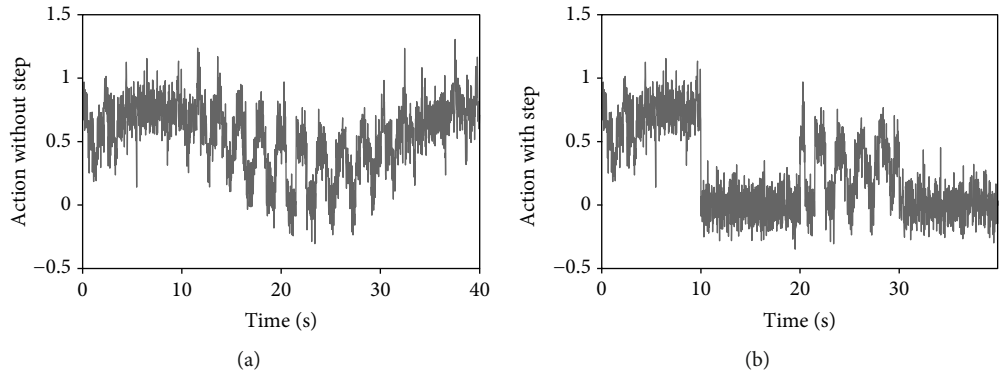

Figure 3: Accelerator pedal signal without (a) and with step factor (b).

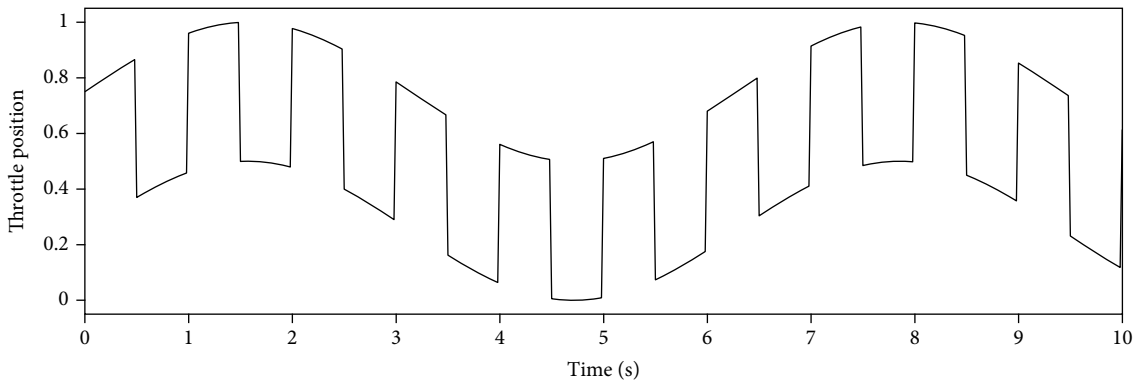

Figure 4: Driver pedal position input.

the controller and a final comparison of the best controllers on four different floors.

3.1. DOE of the Controller. The DOE is performed twice, first, with the controllers including the vehicle velocity, and the evaluation is repeated with the actor network which does not consider the vehicle velocity. The results of the DOE including the vehicle velocity in the state vector do not present a statistical influence on the average reward, while the amount of data present a strong influence on the training time. Figure 5 shows the cube plots of the DOE results for the controllers trained with vehicle velocity included in the state vector.

Evaluating Figure 5(a), the results of the average reward present a small variance. These results indicate a robust application of the training process since the data collection does not significantly influence the controller performance. In this way, no special care is necessary during the data generation, and a reduced quantity of data can be applied to the training process.

Regarding the training time results present in Figure 5(b), the increase of data from 25 to 50 minutes increases the training time by 40 minutes, revealing a nonlinear behavior between the amount of the data and the training time. The smallest time was present with the training process of no step, 0 of reward in bad situations, and 25 minutes of data collection with a total of 7 minutes of training, while the opposite corner showed a training time of 54 minutes.

Since all controllers are already trained and the controller performance is the most important variable in this study, the controller to the next evaluations is chosen just based on the best reward. In Figure 5(a), the best fitted average reward is obtained with data composed by steps, 50 minutes of data, and punishment of -1 as reward. However, in the real data, the best result occurs with the controller trained with null rewards that result in an average reward of 0.877. This behavior happened due to the approximations of the results shown in the DOE by a function and the small influence of the reward.

The results of the controllers which do not apply the vehicle velocity as part of the state vector show statistical influence from step and reward value on the average reward. It indicates a dependence on the data quantity on the controller ability. Furthermore, less difference between rewards and punishments shows more effective results in this approach. Analyzing the cube plot Figure 6(a), the best controller is given by the training process with a greater amount of data, null reward, and data generated without the presence of steps in the throttle pedal position.

The effects on the training time of the controller without the vehicle velocity as an input exhibit significant influence from the amount of data used to train the controller. Comparing the cube plots of both controllers regarding the training time in Figures 5(b) and 6(b), the behavior is similar. Both graphs present the fastest convergence with the training process without steps in the input, a smaller amount of data, and the null reward. In both cases, the slowest converge

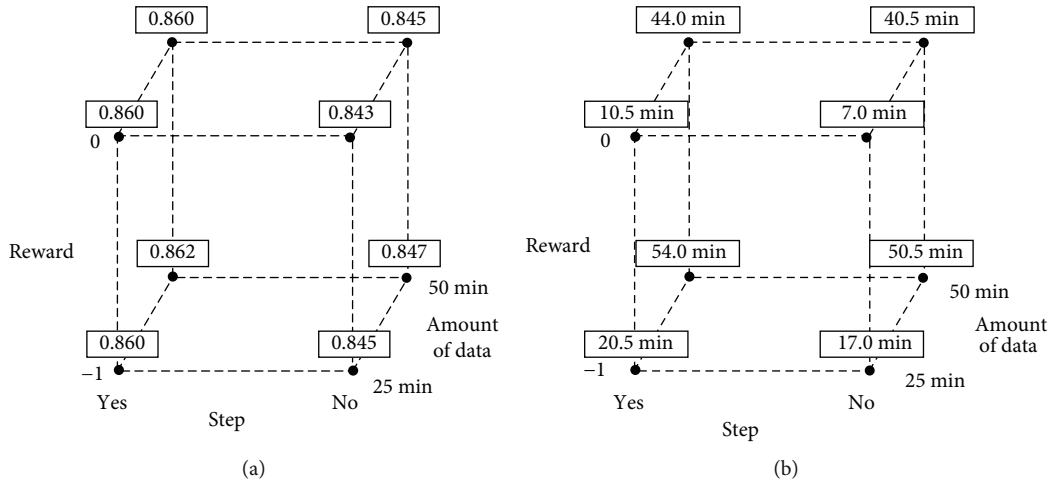

Figure 5: Cube plot of the factors influences on the (a) average reward and (b) time with the vehicle velocity included in the state vector.

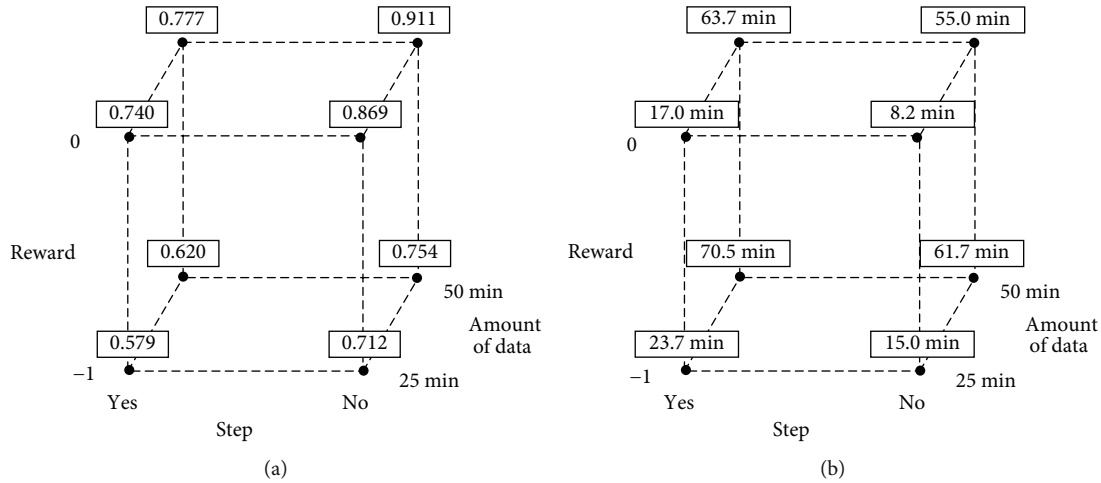

Figure 6: Cube plot of the factors influences on the (a) average reward and (b) the time without vehicle velocity in the state vector.

occurs in the opposite corner of the cube. However, the network without the velocity presents a higher time to converge in the majority of cases.

The different influences of the dataset observed on the performance of the different controllers indicate a necessity for evaluation and redefinition of data every time that the controller architecture is changed, limiting the establishment of standardization of the dataset.

Since all the controllers are already trained and available, the training time was not considered to choose the controller for the next evaluations. In this way, the applied controller without vehicle velocity is trained with 50 minutes of data collection, null reward, and data generated without the presence of steps in the throttle pedal position.

3.2. Comparison of the Controllers. The comparison of the chosen controllers and the vehicle without the antislip control is presented in Figure 7. The left side graphs represent the output derived from the controller when the input from Figure 4 is implemented as the accelerator pedal provided by the driver. The right-side graphs show the slip ratio obtained during the maneuvers.

In the first graph, the absence of a controller generates a desired throttle position identical to the input while the slip ratio increases with the reduction of the friction coefficient. As expected, ice and snow generate high slip ratios which significantly reduce vehicle handling and safety.

With the controller without the vehicle velocity as input, the slip ratio during the trained floors reduced significantly, frequently keeping the slip ratio below the desired value of 0,2. The control behavior can also be checked in the desired acceleration, which is strongly reduced when compared with the input. However, in the nontrained floors, (wet asphalt and ice) the results presented inadequate outputs, with high slip ratios and maintenance of the input throttle pedal magnitude. As presented by Funk Drechsler et al. [1], the lack of the velocity parameter also includes a high-frequency noise mainly in the trained grounds.

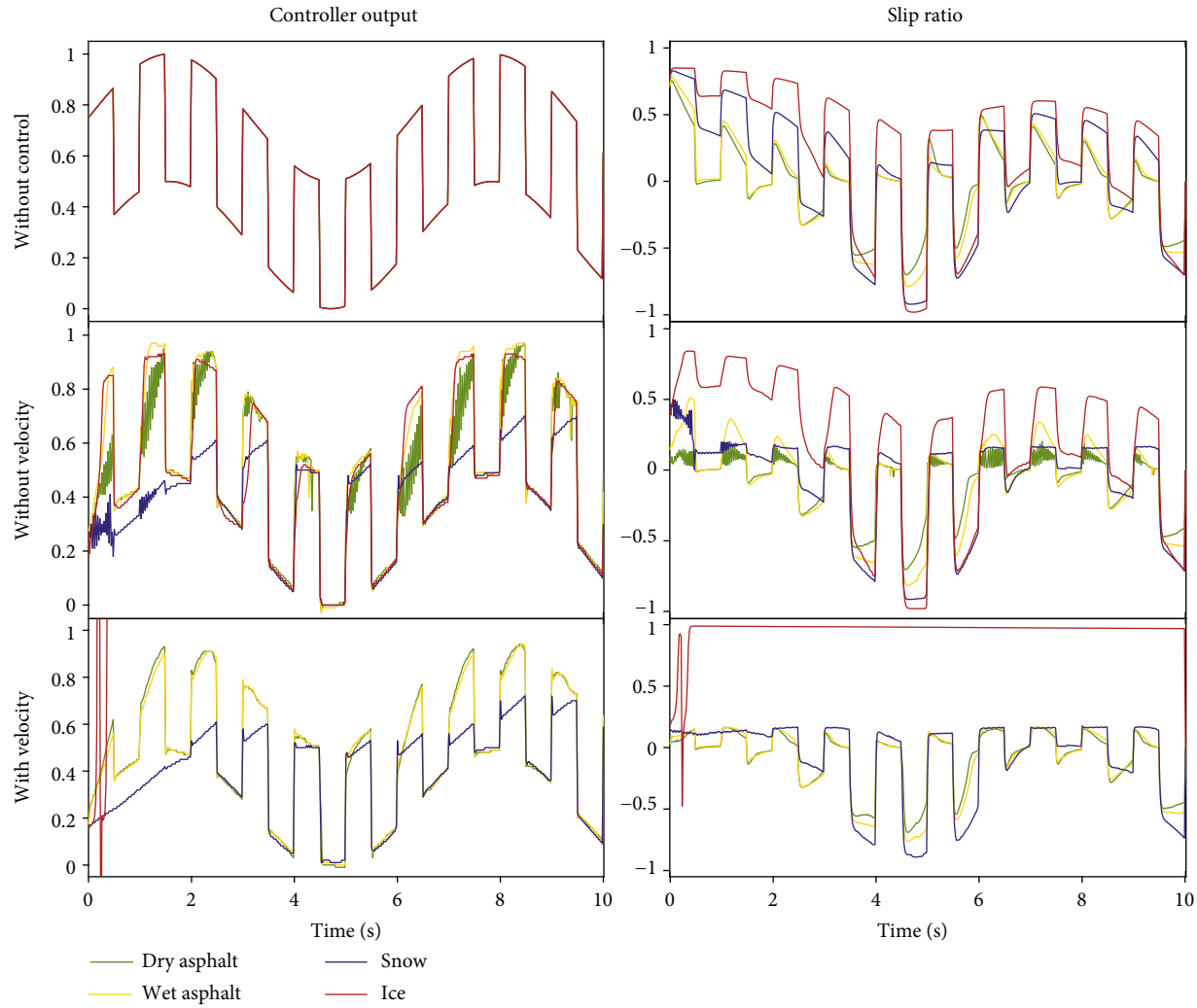

Figure 7: Comparison of the chosen controllers.

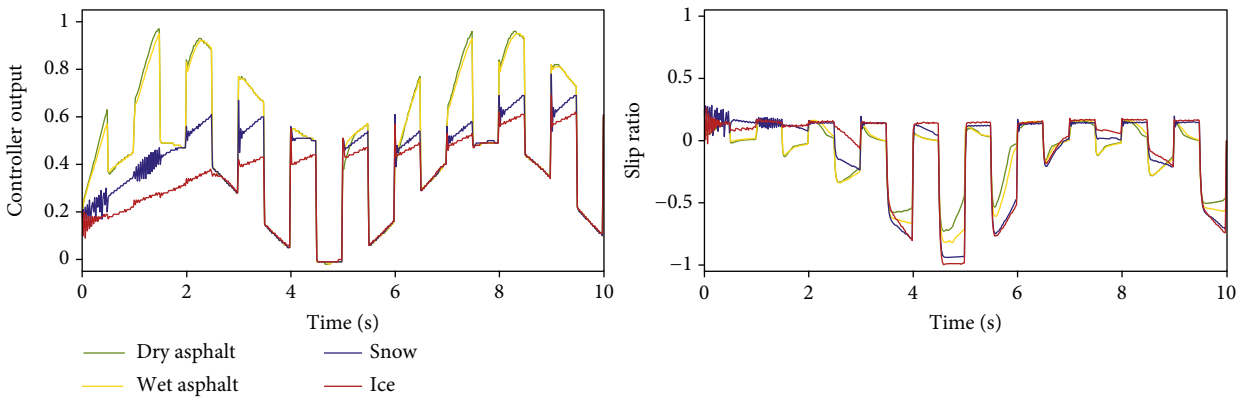

Figure 8: Evaluation of the controller trained on ice and dry asphalt.

The RL controller with vehicle velocity as input parameter presents an excellent behavior on asphalt and snow. On these floors, the slip ratio is maintained always under the desired limit, and the controller can learn the adequate and smooth throttle position variation that maintains the slip ratio near to the maximum acceptable value. In both cases (with and without velocity), it is possible to verify that the network can identify the ground difference even no floor characteristics are directly given to the controller.

However, the ice ground presents a complete unexpected behavior in which the controller that considers the vehicle velocity generates infinity acceleration and maximum slip

during the whole test. This conduct can occur due to the lack of similar states during the training. In this way, a second training process is realized replacing the snow data with the ice data. Thus, dry asphalt and ice floors are applied to the training expecting that the controller can interpolate the road-tire interaction in other road conditions. The training parameters are the same which are applied to the controller with vehicle velocity evaluated in Figure 7.

The obtained result is presented in Figure 8. The proposed controller can keep the desired result in all evaluated floors, maintaining the slip ratio most of the time lower than 0.2. The network can apply different inclinations of the controller output during the time, maintaining the slip ratio constant at 0.2.

This interpolation ability can significantly improve the training process due to the application of extreme maneuvers to collect training data, reducing the necessary amount of data. In the same way, the application of open-loop training algorithms can facilitate the application of already collected data and the collection of data without the necessity of powerful training stations coupled to the moving system. Furthermore, an open-loop method significantly reduces the risks related to the implementation of the network on the final plant.

4. Conclusions

Based on the results of previous researches, this work proposed an open-loop method for training a neural controller to control the slip of the wheels of an electric vehicle. The controller obtained by the reinforcement learning method presents satisfactory control of the vehicle traction even with the nonlinearities of the tires. The use of open-loop training enables easy and safe generation of the data in the real vehicle combined with offline training possibility. It can reduce unexpected and unsafe outputs from the controller during the training phase. Furthermore, the open-loop training enables the acquisition in the vehicle and the training in external servers, removing the necessity of high computational power in the vehicle.

The use of the real vehicle velocity in addition to wheel speed as input to the controller is also evaluated. The absence of the vehicle velocity as controller input shows restrictions in case different grounds are evaluated; however, the application of this architecture seems promising if a larger dataset is implemented in the training process. The sensitivity of this controller with the data variation can also be highlighted since the data quality and quantity influence this architecture more significantly than in the architecture with the vehicle velocity as one of the controller inputs.

On the other hand, the network which considers the vehicle velocity can comprehend the difference between the floors and maintain an adequate slip of the wheels. The network also presents itself as a reliable controller which does not strongly depend on the training data. However, when snow and dry asphalt are implemented in the training process, the network is not able to control the traction of the vehicle on the icy floor. In another way, in case that the ice and dry asphalt are implemented during the training process, the controller keeps the slip in a desirable range in all evaluated floors. It indicates a possible interpolation characteristic in which the training of the extreme conditions (asphalt and ice) permits adequate control on different floors inside this range.

In this way, the proposed RL-based controller can identify the different ground characteristics and apply different throttle positions according to each different floors. Furthermore, it maintains the slip ratio near to the maximum, increasing the friction coefficient of the tire-ground correlation in the longitudinal direction without the degradation of the lateral maximal force.

The current research needs to be further developed to validate the stability of the controller and tests with data collected from the real vehicle need to be performed. In future researches, the authors indicate to define and apply a validation method for the stability of the controller. Furthermore, data from the real vehicle shall be collected, and the obtained controller shall be implemented on the real plant. During this implementation, the influence of the data noise, and the applicability of filtering shall be analyzed. The implementation of the proposed method can also be evaluated for controlling other systems as brake-by-wire, or steering-by-wire.

Data Availability

No underlying data supporting the results of this paper is available.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank AWARE (Applied NetWork on Automotive Research and Education) program from Technische Hochschule Ingolstadt, DAAD (German Academic Exchange Service), FAPESC (Fundação de Amparo à Pesquisa e inovação do Estado de Santa Catarina), and the German state of Bavarian for the financial support during the development of this research.

References

[1] M. Funk Drechsler, T. A. Fiorentin, and H. Göllinger, "Actor-critic reinforcement learning to traction control of an electrical vehicle," in *XXII Encontro Nacional de Modelagem Computacional*, Juiz de Fora, 2019.

[2] U. Samarth and Y. Rashad, "The growth of the automobile industry : Toyota's dominance in united states," *Journal of Research in Marketing*, vol. 3, no. 2, pp. 265–268, 2014.

[3] A. de Oliveira, T. A. Fiorentin, and M. G. Silva, *Design of System and Components - NVH view - Test and Simulation*, SAE Technical Paper, 2008.

[4] C. Bailo, K. Dziczek, B. Smith, A. Spulber, Y. Chen, and M. Schultz, "The great divide: what automotive consumers are buying vs. auto & supplier investments in future

technologies, products & business models," *Center for Automotive Research*, vol. 1, pp. 1–3, 2018.

[5] O. M. Silva, T. M. Guesser, T. A. Fiorentin, and A. Lenzi, "Shape optimization of compressor supporting plate based on vibration modes," *Noise Control Enginnering Journal*, vol. 63, no. 1, pp. 49–58, 2015.

[6] I. Vincent and Q. Sun, "A combined reactive and reinforcement learning controller for an autonomous tracked vehicle," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 599–608, 2012.

[7] M.-B. Radac and R.-E. Precup, "Data-driven model-free slip control of anti-lock braking systems using reinforcement Q-learning," *Neurocomputing*, vol. 275, pp. 317–329, 2018.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, 1st edition, 2007.

[9] M. Wiering and M. van Otterlo, *Reinforcement Learning State-of-the-Art*, Springer, Berlin, 1st edition, 2012.

[10] A. Nandy and M. Biswas, *Reinforcement Learning*, Apress, New York, 1st edition, 2018.

[11] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal on Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[12] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Thesis (Doctor) University of Cambridge, Cambridge, 1989.

[13] M. E. Harmon, L. C. Baird III, and A. H. Klopf, "Reinforcement learning applied to a differential game," *Adaptive Behavior*, vol. 4, no. 1, pp. 3–28, 1995.

[14] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing Atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[15] A. Taitler and N. Shimkin, "Learning control for air hockey striking using deep," 2017, https://arxiv.org/abs/1702.08074.

[16] J. R. de Amaral, H. Göllinger, and T. A. Fiorentin, *Improvement of Vehicle Stability Using Reinforcement Learning*, Anais do XV Encontro Nacional de Inteligência Artificial e Computacional, Ingolstadt, 2018.

[17] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.

[18] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, "Temporal logic motion control using actor–critic methods," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1329–1344, 2015.

[19] D. Zhao, Z. Xia, and Q. Zhang, "Model-Free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [Research Frontier]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 2, pp. 56–69, 2017.

[20] M. Jaritz, R. D. Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, 2017.

[21] P. Wang, C. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1379–1384, Changshu, 2018.

[22] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," in *30th Conference on Neural Information Processing Systems*, Barcelona, 2016.

[23] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 19, pp. 70–76, 2017.

[24] S. Heim, F. Ruppert, A. A. Sarvestani, and A. Spröwitz, "Shaping in practice: training wheels to learn fast hopping directly in hardware," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5076–5081, Brisbane, 2018.

[25] H. Braess and U. Seiffert, *Handbook of Automotive Engineering*, SAE International, Pennsylvania, 2005.

[26] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine Learning*, vol. 84, no. 1-2, pp. 137–169, 2011.

[27] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariante shift," 2015, https://arxiv.org/abs/1502.03167.

[28] "MathWorks, "trainlm,"," 2019, https://de.mathworks.com/help/deeplearning/ref/trainlm.html.