2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Deep Reinforcement Learning for Predictive Longitudinal Control of Automated Vehicles

Martin Buechel[1] and Alois Knoll[2]

Abstract— **This paper presents a predictive controller for longitudinal motion of automated vehicles based on Deep Reinforcement Learning. It uses advance information about future speed reference values and road grade changes. The incorporation of this information leads to a new design parameter with a high influence on learning speed: the selection of proper advance knowledge signals during training. We propose a design method which shows improved learning performance in our experiments. The performance of our controller is explored through simulation of a real world driving scenario in a parking garage. We demonstrate that our Reinforcement Learning agent can learn a behavior close to the optimal solution of a Nonlinear Model Predictive Controller, but at reduced computational costs.**

I. Introduction

Although several solutions exist to solve the problem of longitudinal control of automated vehicles, there are still ongoing efforts to improve control accuracy within the research community (e.g. [1]–[3]). These efforts try to overcome the limitations of classical control solutions, like Proportional-Integral (PI) controllers, which are easy to implement and computationally cheap, but perform poorly regarding control accuracy, comfort, and fuel consumption, especially in presence of disturbances and parameter variations, like road grade changes or a changing vehicle mass [4], [5]. Automated vehicles, especially when connected to intelligent infrastructure (see for example the Providentia project [6]), will have access to considerably accurate advance information about future speed demand values, arising from better planning capabilities.

Many existing control solutions are tailored to the use case of cruise control or adaptive cruise control and can only be applied to traffic scenarios with presence of a lead vehicle ([7]–[15]). For highly automated driving (see [16]), a solution is needed which covers the full range of driving scenarios. In [5], we already proposed an adaptive, Nonlinear Model Predictive Control (NMPC) scheme and showed that it is possible to increase control accuracy by including the advance knowledge about desired trajectories as well as future road grade changes. Recent studies [17] show that the computational complexity of solving the optimal control problem in the NMPC approach still poses challenges for real world applications, when long prediction horizons arise. We desire to incorporate advance knowledge of several seconds in combination with small sample times, which results in a prediction horizon in the order of 20 time steps.

[1]Martin Buechel is with fortiss GmbH, Munich, Germany.
[2]Alois Knoll is with the Department of Robotics, Artificial Intelligence and Embedded Systems, Technical University of Munich (TUM), Munich, Germany

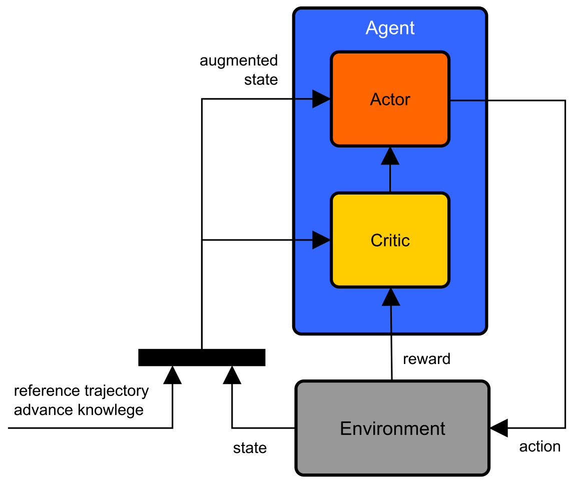

Fig. 1. The Predictive Reinforcement Learning Controller architecture enables the incorporation of advance knowledge about desired reference values, as well as expected disturbances.

Another drawback of model predictive control schemes is the necessity to accurately determine the model parameters, since the robustness to parameter deviations is limited. The identification of these parameters also poses challenges to the practitioner. Other researchers also started working in a direction of model-free longitudinal vehicle control in order to avoid the determination of vehicle model parameters [1].

In recent years, significant progress has been made on Reinforcement Learning (RL) algorithms [18] to solve continuous control problems. The very recent success of applications of Deep Reinforcement Learning on problems like playing the game of Go [19], or learning to play chess [20], was perceived as a breakthrough in artificial intelligence. Within Reinforcement Learning, there is a family of model-free algorithms, which learn a (near-optimal) policy by letting an agent interact with the environment, without the need of prior modeling or parameter identification. There is evidence that RL algorithms are capable to find a close-to-optimal solution compared to Model Predictive approaches [21].

This paper investigates the application of a model-free Deep Reinforcement Learning algorithm on the problem of predictive longitudinal control of an automated vehicle, as an alternative to Model Predictive Control. We analyze the incorporation of advance knowledge into that approach, which leads to the open question of how to design the trajectories of advance information during the learning phase. We propose a method to design advance knowledge trajectories and show that it's application improves learning speed.

The paper is organized as follows: We first give an overview over related work in Section II, explain the problem formulation and the proposed algorithm in Sections III and IV, before presenting simulation details in Section V and results in Section VI. Section VII finally gives a conclusion and outlook on future research.

II. Related Work

We first want to look into applications of Reinforcement Learning in the automated vehicle domain, and then compare various applications of RL for continuous control.

Sallab et al. [22] propose to apply Deep Reinforcement Learning for the end-to-end learning of automated driving, creating actuator commands based on raw sensor inputs. An end-to-end approach was also published in [23], but using supervised learning to train a Convolutional Neural Network to generate steering commands from front camera images.

Mirchevska et al. [24] apply a Fitted Q-iteration algorithm in combination with Extremely Randomized Trees as function approximator to learn the high-level decision making in highway scenarios involving interactions with other road users. They proposed to leave low level commands for steering and acceleration to classic planning and control modules.

Various papers apply RL for (adaptive) cruise controllers. All these methods can only be applied under the assumption of the presence of a lead vehicle. [2] implemented an adaptive longitudinal control method using Neural Dynamic Programming for the acceleration decision while using Internal Model Control for the acceleration tracking. [25] adopted a Policy Gradient RL method to learn a discrete control policy for the adaptive cruise control case. They found that this results in oscillating behavior and propose to extend their algorithm to work in a continuous action space. [26] presents an approach to tune fuzzy controllers using Reinforcement Learning and applies the method to longitudinal vehicle control in a cruise control scenario with a leading vehicle. Due to the nature of fuzzy controllers, it does not exploit information about future vehicle speed or road slope changes. [27] applies Least Squares Policy Iteration to tune the parameters of a PI controller for a cruise controller and validates the proposed solution on an experimental vehicle. The set of possible actions are experimentally predefined combinations of proportional and integral coefficients for different vehicle speed levels. This approach demands a high tuning effort and leaves little room for optimization to the RL algorithm. No predictive information can be incorporated. [28] investigates the application of policy search for learning to control a throttle valve position control using the PILCO algorithm.

Looking at generic Reinforcement Learning algorithms to solve continuous control problems, [29] bench-marked several RL algorithms for continuous control problems. According to their work, the most promising algorithms are the Truncated Natural Policy Gradient (TNPG) [29], the Trust Region Policy Optimization (TRPO) [30], and the Deep Deterministic Policy Gradient (DDPG) algorithm [31]. They found that classical algorithms, like REINFORCE [32] suffer from convergence to local optima.

To the best of the author's knowledge, no application of Deep Reinforcement Learning for the task of the low-level longitudinal vehicle control has been proposed, and hence is regarded as one of the contributions of this paper.

A second contribution is the proposal of a Predictive Reinforcement Learning Controller which not only incorporates information about reference trajectories, but also about future disturbances (see Figure 1). We will discuss that this leads to the question about how to design the advance knowledge trajectories during training of our RL agent, and come up with a proposal. The proposed method is regarded as a third contribution of this paper.

III. Problem formulation

A. Longitudinal vehicle control

We want to find a solution for the task of longitudinal control of automated vehicles, assuming the control module receives reference speed trajectories from a trajectory planning module. We also assume that information about future disturbances regarding road slope changes is available within a prediction horizon of a few seconds and can be exploited in the control module. The longitudinal control module should be capable to send wheel torque demand commands to an actuator module, which actuates on both engine and break. The wheel torque demand will be realized only with a certain delay arising from internal dynamics of the actuators. The reference trajectory information should be incorporated in order to overcome this limitation and hence allows to create smoother control signals to increase comfort and reduce energy consumption.

B. Reinforcement Learning problem

To implement a Reinforcement Learning based controller, we consider a standard RL setup for continuous control in which an agent interacts with an environment E at each discrete time-step t. At each t, the agent gets an observation o_t from the environment, performs an action $u_t \in \mathbb{R}^N$ based on that observation, and receives a scalar reward r_t. We further assume the systems state s_t is fully observed, hence $s_t = o_t$.

The agent's behavior is defined by a deterministic policy π, while the environment E might be stochastic. E is modeled as a Markov Decision Process with an initial state distribution $p(s_1)$, the transition dynamics $p(s_{t+1}|s_t, u_t)$ and the reward function $r(s_t, u_t)$. The goal of a Reinforcement Learning algorithm is to learn a policy $\pi(s, u)$ which maximizes the expected return $\mathbb{E}(\mathrm{R})$ from the start state. The return is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, u_i)$ with the discount factor $\gamma \in [0, 1]$.

IV. Algorithm

We propose to apply a variant of the Deep Deterministic Policy Gradient (DDPG) algorithm [31]. DDPG is an evolution of the Deterministic Policy Gradient (DPG) algorithm [33]. It is a model-free, off-policy algorithm of the family of actor-critic networks [18], which uses Deep Neural Networks (DNN) as function approximators. DDPG allows to learn policies in high-dimensional, continuous state and action spaces. For completeness, a short version of the explanation of the algorithm from the original paper [31] is given here before presenting the algorithm with our modifications:

DDPG uses the action-value function $Q(s,u)$ which denotes the expected return from any state s when performing action u. For a deterministic policy $\mu(s)$ and using the Bellmann equation, Q can be written as

$$Q^{\mu}(s_t,u_t) = \mathbb{E}_{r_t,s_{t+1}\sim E}[r(s_t,u_t)+\gamma Q^{\mu}(s_{t+1},\mu(s_{t+1}))]. \quad (1)$$

Since Q^{μ} is only dependent on the environment, it can be learned off-policy, meaning taking actions from a different policy β. The action-value function Q is approximated by a DNN with parameters θ^{Q}, which can be updated minimizing the loss:

$$L(\theta^{Q}) = \mathbb{E}_{s_t\sim\rho^{\beta},u_t\sim\beta,r_t\sim E}[(Q(s_t,u_t|\theta^{Q})-y_t)^2] \quad (2)$$

where

$$y_t = r(s_t,u_t)+\gamma Q(s_{t+1},\mu(s_{t+1}|\theta^{Q}). \quad (3)$$

In order to stabilize learning, DDPG applies an idea from [34] and uses replay buffers and a separate target network which calculates y_t. Replay buffers act to randomly choose samples out of the history of collected tuples of starting state, action, target state and reward. This is necessary to let the algorithm see independent and identically distributed samples, an assumption of the underlying Markov Chain theory, which does not hold in dynamic environments.

The actor policy $\mu(s|\theta^{\mu})$ is updated using:

$$\nabla_{\theta^{\mu}}J \approx \mathbb{E}_{s_t\sim\rho^{\beta}}[\nabla_u Q(s,u|\theta^{Q})|_{s=s_t,u=\mu(s_t)}\nabla_{\theta_{\mu}}\mu(s|\theta^{\mu})|_{s=s_t}] \quad (4)$$

which is the policy gradient [33]. In order to have the learning components less sensitive to state information of different units and ranges across different environments, a technique called batch normalization [35] is applied. To stabilize Q-learning, copies of the actor and critic network are created and used for calculating the target values (Equation 3). The weights of these target networks are updated to slowly track the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$.

A. State augmentation for predictive control

In order to incorporate advance knowledge to let the agent learn a predictive control policy, the state vector s is augmented with the reference trajectory vector $P=[\rho_t,\dots\rho_{t+N_p}]$, with reference state information over the prediction horizon N_p, and advance knowledge vectors $A_i=[\alpha_t,\dots\alpha_{t+N_a}]$, containing information over the advance knowledge horizon N_a.

B. Choice of excitation signal

The task of Reinforcement Learning for predictive tracking control poses an additional design option to the engineer: the choice of a good reference trajectory during Reinforcement Learning. If advance knowledge about expected disturbances is incorporated, another degree of freedom is added. Since this proposal is a novelty, we want to discuss this here and want to show later that the choice of reference trajectories during the learning phase plays a crucial role for learning performance.

One intuitive choice of reference trajectories for learning would be to train our agent on the same scenario on which we want to evaluate it. One might also believe, that our agent will learn to perform well on this single scenario quickly, but perform poorly on a different one which lies outside the state and action space visited during learning, due to extrapolation effects.

In order to perform well in the whole state space, we need to visit these states also during learning. Covering the whole state space during vehicle testing might be a challenge on test tracks. Considering to perform the training on vehicle test beds gives additional possibilities to the design of advance knowledge trajectories.

We suggest to apply a method from nonlinear dynamic system identification [36] which is broadly used in the domain of internal combustion engine identification, and propose to use Amplitude Modulated Pseudo Random Binary Signals (APRBS) [37][38] for the training phase. APRBS are a sequence of steps with arbitrary amplitudes and a varying step length. We propose to modify the original DDPG algorithm to include the generation of APRBS reference trajectories for each episode during learning. Details about the calculation of APRBS reference signals are given in [38].

C. PRLC-A longitudinal vehicle controller

Our target is to learn a policy which lets a vehicle follow a given speed reference trajectory $v^d=[v^d_t,v^d_{t+1},...,v^d_{t+N_p}]$. In order to make it predictive, we first augment the state with information dependent on v^d, but instead of reference speed values, we calculate $P=[e_t,e_{t+1},\dots,e_{t+N_p}]$ with $e_i=v^d_i-v_t$, which is the deviation of desired speed values from the actual vehicle speed at each time step over the prediction horizon.

We further additionally include advance knowledge information about current and future road slope. For simplicity we keep the advance knowledge horizon N_a identical to the prediction horizon N_p, hence the observations are defined as $o_t=[v_t,a_t,e_t,e_{t+1},\dots,e_{t+N_p},\varphi_t,\varphi_{t+1},\dots,\varphi_{t+N_p}]$, with the vehicle's acceleration a_t at time t and the road grade information φ over the prediction horizon. We call our controller the "Predictive Reinforcement Learning Controller with Incorporated Advance Knowledge (PRLC-A)".

D. Reward function

To reduce the steady state error of the resulting policy, we use a non-quadratic reward function [39] instead of a common quadratic reward function. We also want to penalize high control output values, so we define

$$r(s_t, u_t) = -(q|v_t^d - v_t| + p|u_t|). \tag{5}$$

E. Proposed algorithm

The resulting Algorithm 1 is an enhancement to the DDPG algorithm from [31].

Algorithm 1

```
1:  Randomly initialize critic network $Q(s,u|\theta^Q)$ and
    actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
2:  Initialize target network $Q'$ and $\mu'$ with weights
        $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
3:  Initialize replay buffer $R$
4:  for episode = 1, M do
5:      Initialize a random process $\mathcal{N}$ for action exploration
6:      Create reference trajectory signal using APRBS
        Generator
7:      Receive initial observation state $s_1$
8:      for t = 1, T do
9:          Select action $u_t = \mu(s_t|\theta^\mu) + \mathcal{N}$ according to
            the current policy and exploration noise
10:         Execute action $u_t$ and observe reward $r_t$ and new
            state $s_{t+1}$
11:         Store transition $(s_t, u_t, r_t, s_{t+1})$ in $R$
12:         Sample a random minibatch of $N$ transitions
            $(s_i, u_i, r_i, s_{i+1})$ from $R$
13:         Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
14:         Update critic by minimizing the loss:
                $L(\theta^Q) = \frac{1}{N}\sum_i (y_t - Q(s_t, u_t|\theta^Q))^2$
15:         Update the actor policy using the sampled policy
            gradient:
                $\nabla_{\theta^\mu} J \approx$
                $\frac{1}{N}\sum_i \nabla_u Q(s,u|\theta^Q)|_{s=s_i,u=\mu(s_i)} \nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s_i}$
16:         Update the target networks:
                $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
                $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
17:     end for
18: end for
```

V. Simulation details

To validate the proposed controller, real data from a driving scenario in a parking garage was taken as reference to feed a simulation. The vehicle should follow a speed profile (Figure 2) with different speed levels coming from a planning module, in which higher speeds are requested for longer straight, open passages and speed is reduced to drive around corners or in tight passages.

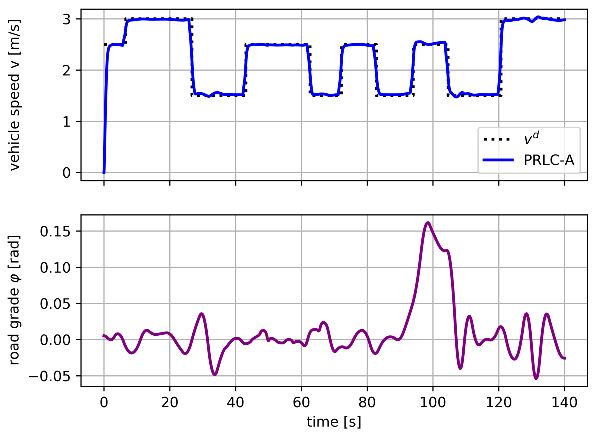

Fig. 2. Evaluation of a Predictive Reinforcement Learning Controller in a real world example. The reference speed trajectories are taken from a driving scenario in a parking garage. The Predictive Reinforcement Learning Controller with incorporated advance knowledge about future road grade changes (PRLC-A) performs well in tracking the desired vehicle speed v^d.

A. Simulation environment

According to [5], the discrete vehicle speed dynamics is modeled with the input $u_t = M_t^d$ as

$$v_{t+1} = T\,\mathrm{g}\big(\mathrm{h}(u_t), v_t, \varphi_t, m\big) + v_t \tag{6}$$

with sample time T, road grade φ and vehicle mass m. Function h describing the first order torque response M_w to a torque demand M^d due to the internal engine dynamics is given as

$$\begin{aligned} M_{w,t} &= \mathrm{h}(M_t^d) \\ &= \frac{1}{\frac{\tau_{e,br}}{T}+1}\left(M_t^d - M_{w,t-1}\right) + M_{w,t-1}. \end{aligned} \tag{7}$$

As in [5], we use a different time constant τ_e for engine and τ_{br} for brake torque build-up.

Function g describes the powertrain dynamics as

$$\begin{aligned} \mathrm{g}(M_{w,t}, v_t, \varphi_t, m) &= \frac{M_{w,t}}{(m + I_{res})\; r_{eff}} \\ &- \frac{1}{m + I_{res}}\Big(mg\left(\sin\varphi_t + C_r\cos\varphi_t\right) - C_d v_t^2\Big) \end{aligned} \tag{8}$$

with I_{res} as the resulting mass from powertrain inertia, r_{eff} the effective wheel radius, C_r the rolling resistance coefficient and C_d the aerodynamic drag coefficient. Further,

$$M_w = \eta_{pwt}\, R_{pwt}\, M_e - M_{br} \tag{9}$$

with powertrain efficiency η_{pwt}, powertrain ratio R_{pwt}, engine torque M_e and break torque M_{br}.

B. Implementation details

The longitudinal vehicle dynamics model according to (6-7) was implemented as OpenAI gym [40] environment in Python. The RL agent was trained with a modified baseline implementation from [41] of Algorithm 1 (in Python and

Tensorflow) which was trained in the following manner: Unless mentioned, our hyper-parameters were identical to the appendix of [31] with DNNs for actor and critic, containing 2 hidden layers with Rectified Linear Units and a tanh layer for the output. Networks were optimized with an Adam optimizer. As proposed in [41], we used 64 units in the hidden layers. The discount factor $\gamma = 0.99$ and for the soft target updates a $\tau = 0.01$ was used. We applied Gaussian noise with $\sigma = 0.02$, since learning performance with parameter noise did not show good learning performance, but this finding was not analyzed any further.

Find the parameters used for our experiments in Table I.

TABLE I
VEHICLE EQUATION SYMBOLS AND SIMULATION PARAMETER

Symbol	Description	[Unit]
v	Vehicle speed	[m/s]
M_w	Wheel torque	[Nm]
M^d	Wheel torque demand	[Nm]
M_e	Engine net torque	[Nm]
M_{br}	Brake torque	[Nm]
φ	Road grade	[deg]
m	Vehicle mass	2000 [kg]
I_{res}	Mass resulting from powertrain inertia	50 [kg]
η_{pwt}	Powertrain efficiency	0.89 [-]
R	Powertrain ratio	8.446 [-]
r_{eff}	Effective wheel radius	0.3 [m]
M_{drag}	Maximum negative engine drag torque	-20 [Nm]
g	Gravitational constant	9.81 [N]
C_r	Rolling resistance coefficient	0.015 [-]
C_d	Aerodynamic drag coefficient	0.4262 [kg/m]
N	Prediction horizon	20
T	Controller sampling time	0.05 [s]
τ_e	Time constant of engine torque response	0.15 [s]
τ_{br}	Time constant of brake torque response	0.05 [s]

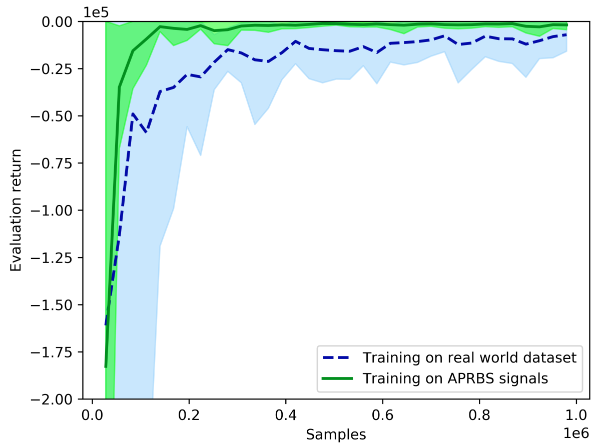

Fig. 3. Comparison of learning speed when the agent trains on different data sets. The learning performance when learning on APRBS signals, as we propose, is considerably better.

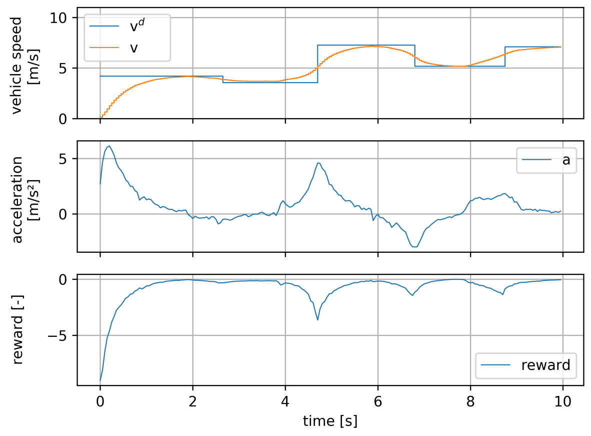

Fig. 4. Vehicle speed response during learning. The predictive policy learns a smooth response behavior to a sequence of short APRBS step responses. The perturbations on the acceleration a are due to action noise and only occur during learning. Please consider the different time scale compared to Figures 2 and 5.

VI. Results

A. Comparison of learning speed

We evaluate the learning performance by training a DDPG agent with reference and advance knowledge information from the real world data set from Figure 2. In other words, we let our agent learn by repeatedly driving through a simulated parking garage, and evaluate the performance on the same driving scenario by calculating the evaluation return.

We then perform training runs on APRBS sequences and evaluate on the parking garage data set as before, and compare the results.

Figure 3 shows the comparison of the learning curves as a mean over 20 runs with a total of 1M steps with 95% confidence intervals. We can see that the learning speed of our approach using APRBS signals is considerably better than when training on the evaluation data set.

B. Training using APRBS sequence

We can see a typical APRBS sequence of steps of random length and vehicle speed demand values in Figure 4. One can see that the PRLC controller learns well to generate smooth transitions to the step demand values, thanks to the predictive nature. Perturbations on the acceleration signal are due to exploration noise, which is necessary during training. Once we observe the agents performance has converged, we stop training and use the generated policy in a closed loop manner without any action noise.

C. Comparison of computation times

A comparison of computation times to calculate the control action of the NMPC approach from [5] with the PRLC-A controller is given in Table II. Times shown are mean values over 2500 cycles and evaluated on a Laptop with an Intel Core i7-6700HQ CPU with 2.6GHz and 16GB

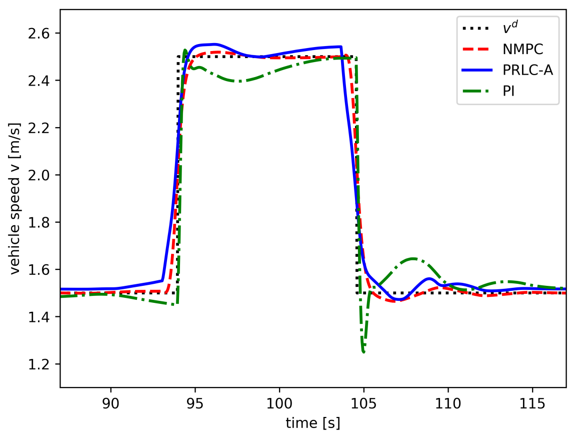

Fig. 5. Comparison of tracking performances of the proposed PRLC-A controller with results from a PI controller and a NMPC controller. The Predictive Reinforcement Learning controller shows a performance close to the optimal solution of the NMPC.

RAM. The NMPC controller was implemented in Matlab and the PRLC-A was inference only with Tensorflow on a CPU, so both were not optimized for fast execution. Although the results are not directly comparable due to implementation in different languages, we find the execution time of the PRLC-A controller is between 30 to 70 times faster, while being almost insensitive to increasing prediction horizons.

TABLE II

RESULTS: EVALUATION OF COMPUTATION TIMES FOR PRLC-A COMPARED TO NMPC WITH DIFFERENT PREDICTION HORIZONS N_p

Controller	N_p**=10**	N_p**=15**	N_p**=20**
PRLC-A	1.112 [ms]	1.112 [ms]	1.113 [ms]
NMPC	37.1 [ms]	57.5 [ms]	81.1 [ms]

D. *Evaluation of PRLC-A controller*

We then evaluated the performance of a resulting controller which was trained on ARPBS signals in comparison to other existing control solutions. First, a standard PI controller was tuned to aperiodic step responses in flat terrain and the relevant speed range, to avoid uncomfortable oscillations. Second, the NMPC controller from [5] was evaluated, which includes identical advance knowledge to calculate the optimal control command.

We evaluate the controllers on the parking garage data set from Figure 2. Figure 5 shows the same evaluation but includes the comparison data. It shows a zoom to the most critical part between seconds 90 and 115 of the data set, when right after climbing up the ramp at increased speed, the vehicle is requested to slow down in order to safely pass a narrow passage at a gate.

We can observe that the PI controller has difficulties to cope with the disturbance due to the ramp, and produces a big undershoot when decelerating. The PRLC-A controller performs close to the optimal solution of the NMPC controller.

VII. CONCLUSION AND OUTLOOK

We demonstrated that Deep Reinforcement Learning has the potential to be used for predictive tracking control which incorporates advance knowledge of disturbances. We applied this technique to low-level longitudinal control of automated vehicles, which are able to provide such information. We found that the design of the trajectories for reference values and advance knowledge has a high impact on learning speed and proposed a method for the design of training experiments. We showed that a Deep Reinforcement Learning based controller, once trained, has considerably low computing times compared to a Nonlinear Model Predictive Controller, while achieving close-to-optimal performance.

For a practical implementation of the proposed approach, many obstacles have to be considered. We found a high variance between different training runs, which has two implications. First, without performing time consuming evaluation runs, it is challenging to choose the best policy. Second, the training time necessary to achieve an acceptable control performance is high, and one can expect that more training samples will be necessary for real world application. We assumed, that no other disturbances or variable parameters, like wind or a changing vehicle mass, are present. Also, no measurement noise was simulated. We want to investigate the robustness of the approach against these influences. To further improve learning performance and stabilize training, we want to look into ideas from Apprenticeship Learning or Imitation Learning.

ACKNOWLEDGMENTS

The German Federal Ministry of Transport and Digital Infrastructure supported this work within the project "Providentia". The financial support is gratefully acknowledged.

REFERENCES

[1] P. Polack, B. D'Andréa-Novel, M. Fliess, A. de la Fortelle, and L. Menhour, "Finite-Time Stabilization of Longitudinal Control for Autonomous Vehicles via a Model-Free Approach," *IFAC-PapersOnLine*, 2017. [Online]. Available: http://arxiv.org/abs/1704.01383

[2] Q. Zhu, B. Dai, Z. Huang, Z. Sun, and D. Liu, "An Adaptive Longitudinal Control Method for Autonomous Follow Driving Based on Neural Dynamic Programming and Internal Model Structure," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, pp. 1–13, 2017.

[3] L. Menhour, B. D'Andrea-Novel, M. Fliess, D. Gruyer, and H. Mounier, "An Efficient Model-Free Setting for Longitudinal and Lateral Vehicle Control: Validation Through the Interconnected Pro-SiVIC/RTMaps Prototyping Platform," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 461–475, 2018.

[4] M. Buechel and A. Knoll, "A Parameter Estimator for a Model Based Adaptive Control Scheme for Longitudinal Control of Automated Vehicles," *9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2016)*, vol. 49, no. 15, pp. 181–186, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316310059

[5] ——, "An Adaptive Nonlinear Model Predictive Controller for Longitudinal Motion of Automated Vehicles," in *2016 IEEE Conference on Control Applications (CCA)*, 2016, pp. 103–108. [Online]. Available: http://ieeexplore.ieee.org/document/7587829/

[6] G. Hinz, M. Buechel, F. Diehl, G. Chen, and A.-k. Kraemmer, “Designing a Far-Reaching View for Highway Traffic Scenarios with 5G-Based Intelligent Infrastructure,” in *8. Tagung Fahrerassistenzsysteme TUEV - Sued*, 2017. [Online]. Available: https://mediatum.ub.tum.de/doc/1421303/1421303.pdf

[7] T. Radke, “Energieoptimale Laengsfuehrung von Kraftfahrzeugen durch Einsatz vorausschauender Fahrstrategien,” Ph.D. dissertation, Karlsruher Institut fuer Technologie (KIT), 2013. [Online]. Available: http://dx.doi.org/10.5445/KSP/1000035819

[8] R. Schmied, H. Waschl, R. Quirynen, and M. Diehl, “Nonlinear MPC for Emission Efficient Cooperative Adaptive Cruise Control,” *IFAC-PapersOnLine*, vol. 48, no. 2014, pp. 160–165, 2015.

[9] D. Corona, M. Lazar, B. De Schutter, and M. Heemels, “A Hybrid MPC Approach to the Design of a Smart Adaptive Cruise Controller,” *Proceedings of the IEEE International Conference on Control Applications*, pp. 231–236, 2006.

[10] T. V. Keulen, G. Naus, and B. D. Jager, “Predictive Cruise Control in Hybrid Electric Vehicles,” *World Electric Vehicle Journal*, vol. 3, pp. 494–504, 2009.

[11] W. Qiu, Q. U. Ting, Y. U. Shuyou, G. U. O. Hongyan, and C. Hong, “Autonomous Vehicle Longitudinal Following Control Based On Model Predictive Control,” in *Proceedings of the 34th Chinese Control Conference*, Hangzhou, 2015, pp. 8126–8131.

[12] D. Zhao, Z. Hu, Z. Xia, C. Alippi, Y. Zhu, and D. Wang, “Full-range Adaptive Cruise Control Based on Supervised Adaptive Dynamic Programming,” *Neurocomputing*, vol. 125, no. 2014, pp. 57–67, 2014.

[13] Z. Xiangrui and W. Junmin, “A Parallel Hybrid Electric Vehicle Energy Management Strategy Using Stochastic Model Predictive Control With Road Grade Preview,” *IEEE Transactions on Control Systems Technology*, 2015.

[14] S. E. Li, Z. Jia, K. Li, and B. Cheng, “Fast Online Computation of a Model Predictive Controller and Its Application to Fuel Economy Oriented Adaptive Cruise Control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1199–1209, 2015.

[15] H.-H. Chiang, “Longitudinal and Lateral Control Design for Vehicle Automated Driving,” Ph.D. dissertation, Department of Electrical and Control Engineering National Chiao Tung University, 2008.

[16] SAE, “SAE Document J3016 - Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems,” 2016. [Online]. Available: https://www.sae.org/standards/content/j3016{_}201609/

[17] S. S. Dughman and J. A. Rossiter, “The Feasibility of Parametric Approaches to Predictive Control when Using Far Future Feed Forward Information,” *IEEE International Conference on Control and Automation, ICCA*, no. 1, pp. 1101–1106, 2017.

[18] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” Cambridge, MA, p. 333, 2017. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antono-glou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M.-l. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the Game of Go with Deep Neural Networks and Tree Search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[20] D. Silver, T. Hubert, J. Schrittwieser, I. Antono-glou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” *arXiv preprint*, pp. 1–19, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[21] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, “Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 517–529, 2009.

[22] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep Reinforcement Learning Framework for Autonomous Driving,” *Electronic Imaging*, pp. 70–76, 2017. [Online]. Available: https://arxiv.org/pdf/1704.02532.pdf

[23] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to End Learning for Self-Driving Cars,” *arXiv preprint*, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[24] B. Mirchevska, M. Blum, L. Louis, J. Boedecker, and M. Werling, “Reinforcement Learning for Autonomous Maneuvering in Highway Scenarios,” Walting, pp. 32–41, 2017.

[25] C. Desjardins and B. Chaib-draa, “Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1248–1260, 2011.

[26] X. Dai, C.-k. K. Li, S. Member, A. B. Rad, S. Member, A. B. Rad, and S. Member, “An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 285–293, 2005.

[27] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen, “Self-learning Cruise Control using Kernel-based Least Squares Policy Iteration,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1078–1087, 2014.

[28] B. Bischoff, D. Nguyen-Tuong, T. Koller, H. Markert, and A. Knoll, *Learning Throttle Valve Control Using Policy Search*. Berlin Heidelberg: Springer, 2013.

[29] Y. Duan, X. Chen, J. Schulman, and P. Abbeel, “Benchmarking Deep Reinforcement Learning for Continuous Control,” *arXiv*, 2016.

[30] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust Region Policy Optimization,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897. [Online]. Available: http://arxiv.org/abs/1502.05477

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous Control with Deep Reinforcement Learning,” *arXiv preprint arXiv:1509.02971*, pp. 1–14, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[32] R. J. Willia, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic Policy Gradient Algorithms,” *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 387–395, 2014.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *arXiv preprint*, pp. 1–9, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[35] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv preprint*, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[36] O. Nelles, *Nonlinear System Identification*. Heidelberg: Springer, 2001.

[37] M. Vogels, E. Martini, K. Gschweitl, P. Mathis, H. Altenstrasser, and M. Buechel, “Dynamic Powertrain Calibration: Using Transient DoE and Modelling Techniques,” *Design of experiments (DoE): In: Engine Development II, Haus der Technik Fachbuch*, vol. 49, 2005.

[38] M. Deflorian and S. Zaglauer, “Design of Experiments for Nonlinear Dynamic System Identification,” *18th IFAC World Con-gress*, pp. 13 179–13 184, 2011. [Online]. Available: http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac11-proceedings/data/html/papers/1502.pdf

[39] J. M. Engel and R. Babuska, “On-line Reinforcement Learning for Nonlinear Motion Control: Quadratic and Non-quadratic Reward Functions,” *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 19, pp. 7043–7048, 2014.

[40] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” pp. 1–4, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[41] P. Dhariwal, C. Hesse, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “OpenAI Baselines,” https://github.com/openai/baselines, 2017.