Formulation and validation of a car-following model based on deep reinforcement learning

Fabian Hart[a], Ostap Okhrin[a,b], Martin Treiber[a,b,*]

[a] *TU Dresden*
[b] *Possible second address*

Abstract

To be written at the end

Keywords: reinforcement learning, car-following model, stochastic processes, string stability, validation, trajectory data

1. Introduction

Autonomous driving technologies are seen as promising solutions to improve road safety, where human errors account for 94% of the total accidents of Transportation National Highway Traffic Safety Administration (2015). Moreover congestion, energy consumption and emissions are intended to be reduced. But autonomous driving is a challenging task since the transportation traffic can be dynamic and unpredictable. On the way to fully autonomous driving, Advanced Driver Assistance Systems (ADAS) has been developed to solve tasks like lane-keeping, lane-changing, car-following, emergency braking, etc. Since Deep Learning methods has been demonstrated to surpass humans in certain domains, they are also adopted in the area of autonomous driving. Especially Deep Reinforcement Learning (DRL), which combines the power of Deep Learning in tackling large, complicated problems with Reinforcement Learning, has shown its potential in a broad variety of autonomous driving tasks. In Wang and Chan (2018) and Lin et al. (2020), DRL is used to guide an autonomous

*Corresponding author
Email address: Martin.treiber@tu-dresden.de (Martin Treiber)
URL: www.mtreiber.de (Martin Treiber)

vehicle safely from on-ramp to freeway. In Isele et al. (2018), Gong et al. (2020) and Tolebi et al. (2018), DRL methods are used to manage traffic of autonomous vehicles at intersections, optimizing safety and efficiency. In Wang et al. (2018), DRL is used to solve lane change maneuvers.

A further task in autonomous driving is to model the vehicle behavior under car following scenarios, where suitable accelerations has to computed in order to achieve a safe and comfortable driving. Approaches to solve this task are classical car-following models, such as the Intelligent Driver Model Treiber et al. (2000) or stochastic car-following models, such as in Treiber and Kesting (2018). Furthermore data-driven approaches use Machine Learning methods to train a car-following model based on experimental car-follower data, such as in Chong et al. (2011) or Zhou et al. (2017). Downside of this approach is that the model tries to emulate human driver behavior which still can be suboptimal.

To overcome this issue, DRL methods train non-human car-following models that can optimize metrics such as safety, efficiency and comfort. One approach is to train on scenarios, where the leading vehicle trajectory, used for training, is based on experimental data, such as in Zhu et al. (2020) or Zhu et al. (2018). Similar approaches suggest a standardized driving cycle, which functions as leading vehicle trajectory, such as Lin et al. (2019) or Yuankai et al. (2019), which uses the New European Driving Cycle. A disadvantage coming along with these approaches is, that for scenarios, which are not in the scope of the training data, the performance decreases, indicating inadequate machine learning generalization Lin et al. (2019).

Another issue of car-following models is string-stability. There are several studies focusing on dampening traffic oscillations by using a sequence of trained vehicles, such as Qu et al. (2020), Kreidieh et al. (2018) and Jiang et al. (2020).

All the mentioned DRL car-following models have three disadvantages in common: At first the acceleration range is limited in a way, that full-brakes are not considered. This results in models that are just designed for non-safety-critical scenarios. Second these models just consider car-following scenarios, while free driving or the transition between both is not reflected in the reward

function. Third the trained models have just been validated on data that is similar to the training data set, so that the generalization capabilities cannot be proved.

This motivated us to design a model, which overcomes these issues. To our knowledge, no RL based car-following model has been proposed which has the following features combined:

- The proposed model considers free driving, car-following, as well as the transition between both in a way, that approaching of the leading vehicle is smooth and comfortable.
- The proposed model has a wider range of possible accelerations, which leads to a collision-free behavior also in safety-critical situations such as full-braking of the leader.
- The proposed model is trained on leading trajectories, based on an AR(1)-process, e. g. Honerkamp (1993), the parameters reflecting kinematics of real drivers. This leads to high generalization capabilities and a model usage in a broader variety of traffic scenarios.
- Different driver characteristics can be modeled by adjusting the parameters of the reward function.
- The proposed model shows string-stability.

Another feature of this work is a thorough validation of the above mentioned properties in scenarios based on both synthetic and naturalistic trajectory data, bringing the model to its limits. In all cases, the model proved to be accident free and string stable. In a further experiment the proposed model is compared to an Intelligent Driver Model, calibrated on the same data. The results indicate a better performance of the proposed model.

[short textual enumeration of the sections to come]

2. Reinforcement Learning Background

The following vehicle is controlled by a Reinforcement Learning (RL) agent. By interaction with an environment, the RL agent optimizes a sequential decision making problem. At each time step t, the agent observes an environment state s_t and, based on that state, selects an action a_t. After conducting action a_t, the RL agent receives a reward $r(a_t, s_t)$. The agent aims to learn an optimal state-action mapping policy π that maximizes the expected accumulated discounted reward

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{1}$$

where $\gamma = (0, 1]$ denotes the discount factor and $\gamma^k r_{t+k}$ the expected contribution k time steps ahead.

2.1. RL algorithm

In various similar control problems, the Deep Deterministic Policy Gradient (DDPG) Algorithm has been used and proven to perform well on tasks with a continuous action and state space, such as in Zhu et al. (2020), Lin et al. (2019) or Zhu et al. (2018). The original work can be found in Lillicrap et al. (2015). DDPG is an Actor-Critic method, that uses an Actor $\mu\left(s \mid \theta^{\mu}\right)$ with weights θ^{μ} to propose an action based on a given state s and a Critic $Q\left(s, a \mid \theta^{Q}\right)$ with weights θ^{Q} to predict if the action is good or bad, based on a given state s and action a. To achieve a better training stability, DDPG uses target networks μ' and Q' for Actor and Critic. While training, these networks are updated slowly, hence keeping the estimated targets stable. Furthermore DDPG uses Experience Replay, that implements a Replay Buffer R, where a list of tuples (s_t, a_t, r_t, s_{t+1}) are stored. Instead of learning from the most recent experience, DDPG learns from sampling a mini-batch from the experience buffer. To implement better exploration by the Actor network, DDPG uses noisy perturbations, specifically an Ornstein-Uhlenbeck process for generating noise. It samples noise from a correlated normal distribution. The entire DDPG algorithm is shown in Algorithm 1.

Algorithm 1: DDPG algorithm according to Lillicrap et al. (2015)

Randomly initialize critic network $Q\left(s, a \mid \theta^{Q}\right)$ and actor $\mu\left(s \mid \theta^{\mu}\right)$ with weights θ^{Q} and θ^{μ}

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$

Initialize Replay Buffer R

for *episode* $= 1$ **to** M **do**

Initialize a random process $\mathcal{N}$ for action exploration

Receive initial observation state s_1

for $t = 1$ **to** T **do**

Select action $a_t = \mu\left(s_t \mid \theta^{\mu}\right) + \mathcal{N}_t$ according to the current policy and exploration noise

Execute action a_t and observe reward r_t and observe new state s_{t+1}

Store transition (s_t, a_t, r_t, s_{t+1}) in R

Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

Set $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$

Update critic by minimizing the loss:

$L = \frac{1}{N} \sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2$

Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s, a \mid \theta^{Q}\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu\left(s \mid \theta^{\mu}\right)\Big|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

2.2. Modularized Reinforcement Learning

Furthermore we used a modularized approach to decompose the task into two subtasks. Modular Reinforcement Learning (MRL) refers to the decomposition of a complex, multi-goal problem into a collection of simultaneously running single goal learning processes, typically modeled as Markov Decision Processes. Typically, these subagents share an action set but have their own reward signal and state space. At each time step, every subagent reports a numerical preference for each available action to an arbitrator, which then selects one of the actions for the agent as a whole to take (Bhat et al. (2006)). There are numerous works, that are using a modularized Reinforcement Learning approach, like in Cai et al. (2021), Wang et al. (2017) or Andreas et al. (2016) to name just a few. The advantage of decomposing multiple-goal reward functions with MRL, we also want to use in this work. Figure 1 shows the architecture of our MRL System. We divide our car-following problem into two subtasks, handled by two different policies. The Free-Driving-Policy refers to free driving and aims to not to exceed a desired speed. The Car-Following-Policy refers to following a vehicle and aims to keep a reasonable gap to a leader vehicle. Although both policies are trained with different reward functions and in different training environments, they both output an acceleration value. As an arbitrator between both accelerations we use a simple min-function. In the next sections the model specifications of the Free-Driving-Policy and the Car-Following-Policy, which are both trained with the DDPG algorithm, are described in detail.

3. Free-Driving-Policy

3.1. Action and state space

The defined modularized RL approach requires that the action space of both sub-policies are identical. To enable comfortable driving and allow for a maximum braking deceleration in safety-critical situations, the acceleration is defined as a continuous variable in the range between $a_{\min} = -9\,\mathrm{m/s^2}$ and $a_{\max} = 2\,\mathrm{m/s^2}$.

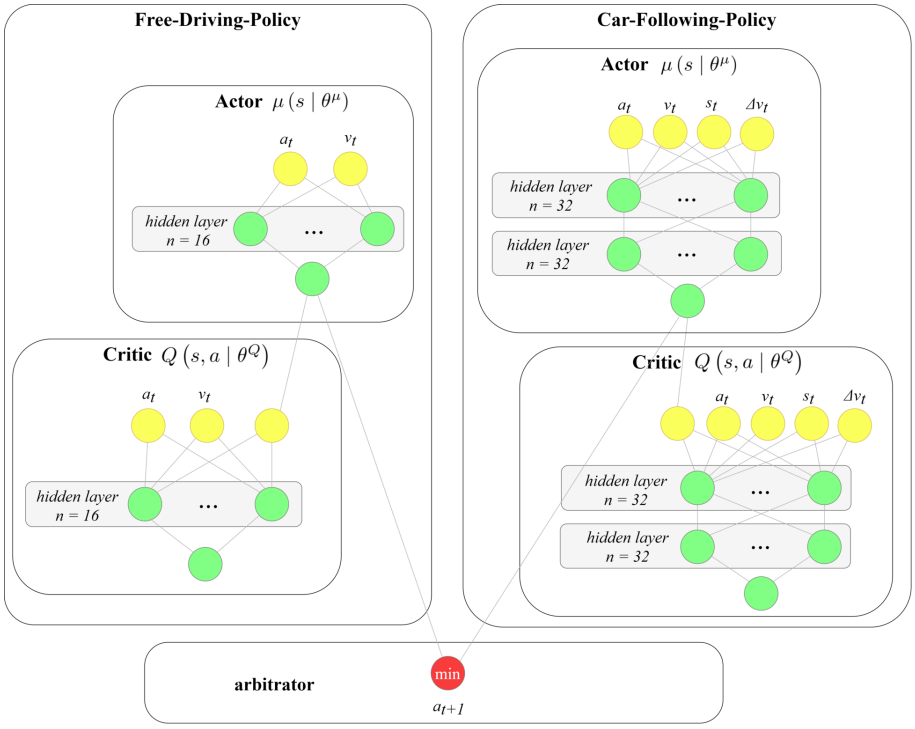

Figure 1: Modularized RL architecture with actor networks of both policies.

The state space defines the observations that the vehicle can receive from the environment. To compute an optimal acceleration, the following vehicle observes its own acceleration a, and its own speed v. Linear translation and scaling is used to reduce the dimensions and to bring all observations approximately into the same range of $[0, 1]$. The observation at time step t is defined as

$$s_t = \begin{pmatrix} \frac{v_t}{v_{\text{des}}} \\ \frac{a_t - a_{\min}}{a_{\max} - a_{\min}} \end{pmatrix}. \tag{2}$$

3.2. Reward Function

The reward function contribution a time step t is decomposed into two factors. The first part aims to to not to exceed a desired speed v_{des}, but also to accelerate if the desired speed is not reached yet.

$$r_1 = \begin{cases} \frac{v}{v_{\text{des}}}, & \text{if } v < v_{\text{des}} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The second part of the reward function aims to reduce the jerk in order to achieve comfortable driving,

$$r_2 = -\left(\frac{1}{\dot{a}_{\text{comf}}} \frac{\mathrm{d}a}{\mathrm{d}t} \right)^2. \tag{4}$$

Since building up a comfortable value of acceleration or deceleration from zero in one second is at the limit of comfortable jerks, r_2 values above unity tend to feel uncomfortable.

The contribution of the final reward function at simulation time step t is the weighted sum of these factors according to

$$r_t = r_1 + w_{\text{jerk}} r_2, \tag{5}$$

where all the factors are evaluated at time step t.

3.3. Training environment

In order to train the RL agent for the task of keeping a desired speed, the training environment is defined as follows. To describe the dynamics of

the vehicle, a point-mass kinematic model is used. For updating the speed and position for time step $t+1$ the Euler and ballistic methods are used, respectively. The roman subscripts ('sub") are only used for textual subscripts ($x_{\max}$, x_{phys}), not for variable subscripts. There, the normal underscore is used

$$v_{t+1} = v_t + a_t d \tag{6}$$

$$x_{t+1} = x_t + \frac{v_t + v_{t+1}}{2} d \tag{7}$$

with d corresponding to the simulation step size, which is globally set to 100 ms. To train the RL agent, a training episode has to be defined. One training episode contains 300 time steps and the vehicle's initial speed is set randomly in the range $[0, v_{\text{des}}]$.

3.4. RL hyperparameters

Both neural networks, critic and actor,[critic and actor are generic concepts, thus lowercase. Named policies (Free-Driving Policy or free-driving policy) may be written upper- or lowercase but always without a hyphen connecting the last word] are feed-forward neural networks with one layer of hidden neurons, containing 16 neurons (see Figure 1). ReLU activation functions are used [define ReLU or give a reference]. The learning rate for Critic and Actor is set to 0.001. For the exploration of the action space, an exploration noise model has to be defined. We adopted a zero-reverting Ornstein-Uhlenbeck process with $\Theta = 0.15$ and $\sigma = 0.2$ as suggested in Lillicrap et al. (2015). The update rate of the target networks is set to 0.001. All DDPG parameters are presented in Table 1.

4. Car-Following-Policy

4.1. Action and state space

To match the action space of the Free-Driving-Policy, the acceleration is analogously defined as a continuous variable in the range between $a_{\min} = -9\,\text{m/s}^2$ and $a_{\max} = 2\,\text{m/s}^2$.

Table 1: DDPG parameter values

	Free-Driving-Policy	Car-Following-Policy
Learning rate	0.001	0.001
Reward discount factor	0.95	0.95
Experience buffer length	100000	100000
Mini batch size	32	32
Ornstein-Uhlenbeck Θ	0.15	0.15
Ornstein-Uhlenbeck σ	0.2	0.2
Number of hidden layers	1	2
Neurons per hidden layer	16	32
Soft target update τ	0.001	0.001

The state space defines the observations that the following vehicle can receive from the environment. To compute an optimal acceleration, the following vehicle uses its own acceleration a, its own speed v, the leader's speed v_l, and the (bumper-to-bumper) gap g [replaced gap s by gap g, also $s_0 \to g_0, s^* \to g^*$.] Again, linear translation and scaling is used to reduce the dimensions and to bring all observations approximately into the same range of $[0, 1]$. [The third state usually is in $[-1, 1]$ and $g/g_{\max}$ needs a fallback if there is no leader] The observation at time step t is defined as

$$s_t = \begin{pmatrix} \frac{v}{v_{\text{des}}} \\ \frac{a - a_{\min}}{a_{\max} - a_{\min}} \\ \frac{v_l - v}{v_{\text{des}}} \\ \frac{s}{s_{\max}} \end{pmatrix}, \tag{8}$$

where $g_{\max}$ is set to 200 m.

Table 2: RL agent parameters and default values

Parameter	Description	Value
$a_{\min}$	Minimum acceleration	$-9\,\mathrm{m/s^2}$
$a_{\max}$	Maximum acceleration	$2\,\mathrm{m/s^2}$
b_{comf}	Comfortable deceleration	$2\,\mathrm{m/s^2}$
$\dot{a}_{\mathrm{comf}}$	Comfortable jerk	$2\,\mathrm{m/s^3}$
v_{des}	Desired speed	$15\,\mathrm{m/s}$
T	Desired time gap to the leading vehicle	$1.5\,\mathrm{s}$
$g_{\min}$	Desired minimum space gap	$2\,\mathrm{m}$
T_{lim}	Upper time gap limit for zero reward (see Eq. (11))	$15\,\mathrm{s}$
w_{gap}	relative weight for keeping the desired gap	0.5
w_{jerk}	relative weight for comfortable acceleration	0.004

4.2. Reward Function

The goal of the Car-Follower-Policy is to reduce the crash risk, while maintaining comfortable driving in non-safety-critical situations. The reward function includes a set of parameters that can be adjusted to realize different driving styles.

The reward function contribution at time step t consists of three factors. The first factor addresses the driver's response in safety-critical situations by comparing the kinematically needed deceleration (assuming an unchanged speed of the leader) with the comfortable deceleration b_{comf}, [a theta function $\Theta(v - v_l) = \Theta(-v_l - v)$ is missing in Eq. (10)] [Why a tanh function? (9) has a kink at zero and flattens if the situation becomes really critical. Why not $-((b_{\mathrm{kin}} - b_{\mathrm{comf}})/a_{\min})^2\Theta(b_{\mathrm{kin}} - b_{\mathrm{comf}})$ possibly normalized to a max of 1 if this is needed?]

$$r_1 = \begin{cases} -\tanh\left(\frac{b_{kin} - b_{\mathrm{comf}}}{-a_{\min}}\right), & \text{if } b_{kin} > b_{\mathrm{comf}} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

with

$$b_{kin} = \frac{(v_l - v)^2}{g} \tag{10}$$

The argument of the tanh function with the maximum possible deceleration ($\approx 9\,\mathrm{m/s^2}$ on dry roads) gives a non-dimensional measure for the seriousness of the critical situation with values near or above 1 indicating an imminent crash. The tanh function functions as a limitation for the reward value to the range [-1,0]. This has shown to make the learning process more stable. Notice that the case distinction in (9) ensures that this term is not activated in non-critical situations.

The second part of the reward function aims to not fall below a reasonable distance to the leading vehicle. [cf. my comment in the chat of the meeting May 26. There, I suggested Erlang functions $r(x) = x^n \exp(-\lambda x)$, e.g., $x \exp(-\lambda x)$. with $1/\lambda = s - g_{\mathrm{opt}}$. The fact that it is constant instead of linearly decreasing for large gaps does not matter since then the free-traffic regime takes over, anyway]

$$r_2 = \begin{cases} \frac{\varphi((g-g_{\mathrm{opt}})/g_{\mathrm{var}})}{\varphi(0)}, & \text{if } g < g^* \\ \frac{\varphi((g-g_{\mathrm{opt}})/g_{\mathrm{var}})}{\varphi(0)} \left(1 - \frac{g-g^*}{g_{\mathrm{lim}}-g^*}\right) & \text{otherwise} \end{cases} \tag{11}$$

with

$$g_{\mathrm{opt}} = vT + g_{\mathrm{min}}, \tag{12}$$

$$g_{\mathrm{var}} = 0.5 g_{\mathrm{opt}}, \tag{13}$$

$$g_{\mathrm{lim}} = vT_{\mathrm{lim}} + 2g_{\mathrm{min}}, \tag{14}$$

and $\varphi(x)$ describing the standard-normal probability density function. The value of g^* is chosen in a way, that the reward function r_2 is differentiable. Figure 2 illustrates the reward function for r_2, containing the parameter g_{opt}, g^* and g_{lim}. The reward function is designed in a way, that for high speeds v of the following vehicle the time gap between following and leading vehicle tends to T, while for low speeds the distance between both tends to g_{min}. Different

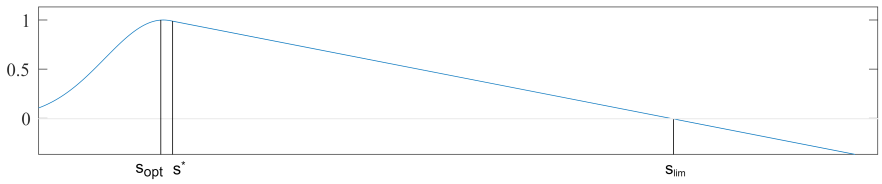

Figure 2: Factor 2 of the reward function maximizing the reward for car following with time gap T

values of T result in different driving styles in a way that, for lower values of T and $g_{\min}$, the driver follows more closely the leading vehicle resulting in a more aggressive driving style. The results for different values of T can be found in Section 5.4. Different functions for $g > g^*$ has been applied, but the best results regarding a smooth and comfortable approaching of the following vehicle has been reached with a linear function. Also, a high value of $T_{\lim}$ results in an early deceleration and comfortable approaching.

The third factor of the reward function aims to reduce the jerk in order to achieve comfortable driving and has been designed analogously to the Free-Driving-Policy,

$$r_3 = -\left(\frac{1}{\dot{a}_{\text{comf}}}\,\frac{\mathrm{d}a}{\mathrm{d}t}\right)^2. \tag{15}$$

The contribution of the final reward function (1) at simulation time step t is the weighted sum of these factors according to

$$r_t = r_1 + w_{\text{gap}} r_2 + w_{\text{jerk}} r_3, \tag{16}$$

where all the factors are evaluated at time step t. The weights (cf. Table 2) have been found experimentally and can be optimized in future studies.

A non-emergency relative-speed term is missing, e.g., $r_4 = -v_{\text{des}}/b_{\text{comf}}(v - v_l)\theta(v - v_l)/g$

4.3. Training environment

The training environment is modeled by a leader and a follower vehicle. Both vehicles implement the point-mass kinematic model described in Section 3.3.

While the follower vehicle is controlled by the RL agent, the leading trajectory is based on an AR(1) process, whose parameters reflect the kinematics of real leaders. The AR(1) process describes the speed of the leading vehicle and is defined as

$$v_l(t) = c + \phi v_l(t-1) + \epsilon_t \tag{17}$$

with

$$E(\epsilon_t) = 0,\ \mathrm{Var}(\epsilon_t) = \sigma,\ \mathrm{Cov}(\epsilon_t \epsilon_{t+k}) = 0 \text{ for } k \neq 0 \tag{18}$$

After reaching stationarity, this process has

$$\text{the expectation value } E(v_l) = \frac{c}{1-\phi}, \tag{19}$$

$$\text{the variance } \mathrm{Var}(v_l) = \frac{\sigma^2}{1-\phi^2}, \tag{20}$$

$$\text{the autocorrelation } \mathrm{ACF}(\Delta t) = \phi^{\Delta t}, \tag{21}$$

$$\text{and the correlation time (in multiples of } d) \ \tau = -\frac{1}{\ln \phi}, \tag{22}$$

To adjust the parameters of the AR(1) process, typical values for real leader trajectories has to be defined: With $v_{l,\mathrm{des}}$ as the desired speed for the leader, the mean of the AR(1) process is set to be $v_{l,\mathrm{des}}/2$ and the standard deviation is set to be $v_{l,\mathrm{des}}/2$. The acceleration a_{phys} corresponds to typical physical leader accelerations. Relating the acceleration to the physical correlation time τd via $a_{\mathrm{phys}} = v_{l,\mathrm{des}}/(2\tau d)$ and using Equation (19) - (22), the parameters of the AR(1) process can be calculated as:

$$\phi = \exp\left(-\frac{2a_{\mathrm{phys}} d}{v_{l,\mathrm{des}}}\right), \tag{23}$$

$$c = (1-\phi)\frac{v_{l,\mathrm{des}}}{2}, \tag{24}$$

Table 3: Assumed typical values for leading trajectories and the resulting values of the AR(1) process parameters for an update time step of 100 ms

Real trajectory		AR(1) process	
$v_{l,\mathrm{des}}$	$15\,\mathrm{m/s}$	ϕ	0.9868
a_{phys}	$1\,\mathrm{m/s^2}$	c	$0.0993\,\mathrm{m/s}$
		σ^2	$1.475\,\mathrm{m^2/s^2}$

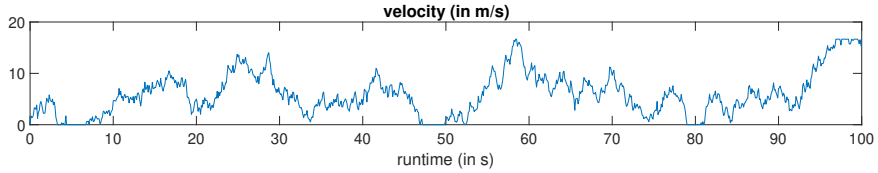

Figure 3: Example of a leading trajectory based on the parametrized AR1 process used to train the RL agent

$$\sigma^2 = (1-\phi^2)\frac{v_{l,\mathrm{des}}^2}{4}. \tag{25}$$

The assumed typical values for $v_{l,\mathrm{des}}$ and a_{phys} as well as the resulting values of the AR(1) process parameters can be found in Table 3. Figure 3 shows an example trajectory of the leading vehicle based on the AR(1) process using the parameters of Table 3. If the speed exceeds the defined range of $[0, v_{l,\mathrm{des}}]$, it is manually set to the range limits. [Consider applying the lower limit, only, as discussed May 26] One episode has a simulation time of 50 s with a step size of $d = 100\,\mathrm{ms}$ resulting in an episode length of 500 steps. The initial speeds of the following and leading vehicles are randomly set in the range $[0, v_{\mathrm{des}}]$ and $[0, v_{l,\mathrm{des}}]$, respectively. The initial space gap between both is set to 120 m.

4.4. RL hyperparameters

Both neural networks, Critic and Actor, are feed-forward neural networks with two layers of hidden neurons, containing 32 neurons each (see Figure 1).

ReLU activation functions are used. We used the same Ornstein-Uhlenbeck model as we used for the Free-Driving-Policy. All DDPG parameters are presented in Table 1.

5. Validation

The goal is not to minimize some error measure as in usual calibration/validation but to check if the driving style is safe, effective, and comfortable. The RL strategy is evaluated with respect to these metrics in different driving scenarios, described in the following.

5.1. Response to an external leading vehicle speed profile

The first scenario is designed in order to evaluate the transition between free driving and car-following as well as the follower's behavior in safety-critical situations. Figure 4 shows a driving scenario with an artificial external profile for the leading vehicle speed. The initial gap between follower and leader is 200 meters, which refers to a free driving scenario. The follower accelerates with $a_{\max} = 2m/s^2$ until the desired speed $v_{\text{des}} = 15\,\text{m/s}$ is reached and approaches the standing leading vehicle. When the gap between both drops below 90 meters, the follower starts to decelerate with a maximum deceleration of approximately $b_{\text{comf}} = 2\,\text{m/s}^2$ (transition between free driving and car-following) and comes to a standstill with a final gap of approximately $g_{\min} = 2\,\text{m}$. Afterwards ($t = 30\,\text{s}$), the leading vehicle accelerates to a speed that is below the desired speed of the follower before performing a maximum braking maneuver ($a = -9\,\text{m/s}^2$) to a full stop ($t = 46\,\text{s}$). At the time of the start of the emergency braking, the follower has nearly reached a steady following mode at the desired space gap $g = s_0 + v_l T$. While this gap makes it impossible to keep the deceleration in the comfortable range without a rear-end collision, the follower makes the best of the situation by braking smoothly with a maximum deceleration of $-a \approx 5\,\text{m/s}^2$. The transition between different accelerations happens in a comfortable way reducing the resulting jerk. Only at the beginning ($t = 46\,\text{s}$)

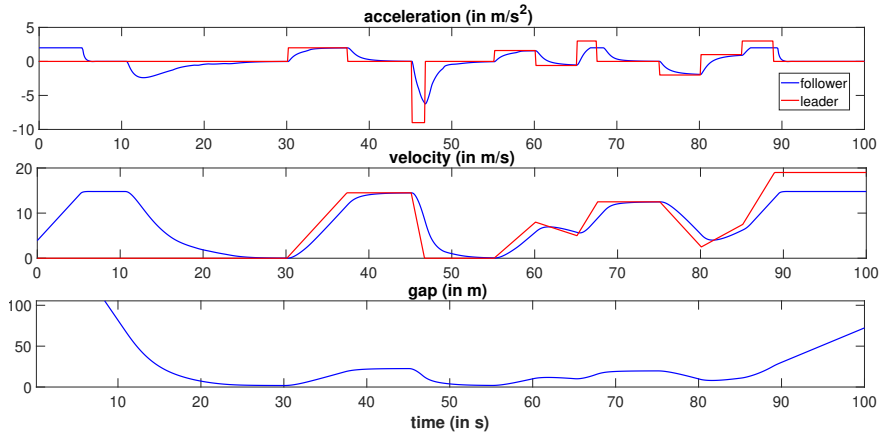

Figure 4: Response to an external leading vehicle speed profile. [Change velocity to speed in all plots]

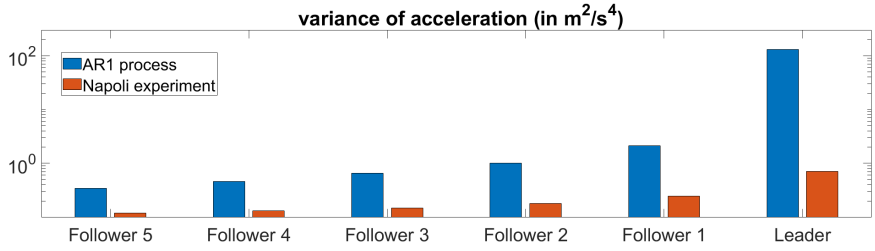

Figure 5: Comparison of the acceleration variance between leader and follower for a leader controlled by AR(1) (blue bars) and the leading vehicle of the Napoli experiment (red).

where the situation is really critical, the jerk $\mathrm{d}a/\mathrm{d}t$ exceeds the comfortable range $\pm 1.5\,\mathrm{m/s^3}$. Afterwards, the leader performs a series of non-critical acceleration and deceleration maneuvers and the follower tries to follow the leader at the speed dependent desired space gap $g_0 + v_l T$ while simultaneously smoothing the leader's speed profile. After the leaders speed exceeds the desired speed of the follower at $t = 88\,\mathrm{s}$ (transition between car-following and free driving), the follower keeps the desired speed $v_{\mathrm{des}} = 15\,\mathrm{m/s}$.

5.2. String stability

The second validation scenario, shown in Figure 6, consists of a leader based on the AR(1) process that is followed by five vehicles, each controlled by the

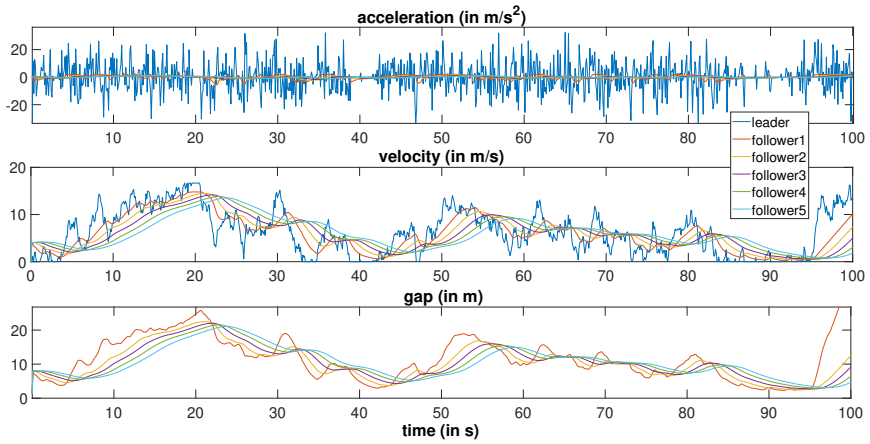

Figure 6: Response to a leader trajectory based on a AR(1) process

trained RL agent. The results show that traffic oscillations can effectively be dampened with a sequence of trained followers, even if the leader shows large outliers in acceleration. Figure 5 illustrates the difference of accelerations between leader and the followers (blue bars). The last follower shows the lowest variance of acceleration, which demonstrates the ability of the RL agent to flatten the speed profile, to dampen oscillations and thus to increase comfort and reduce fuel consumption and emissions.

5.3. Response to a real leader trajectory

In a further scenario, the abilities of the RL strategy are evaluated with a real leader trajectory (Fig. 7). This trajectory comes from platoon driving experiments in Napoli where high-precision distance data between the platoon vehicles were obtained (reference to Punzo et al.). Similar to the experiment from Section 5.2, string stability and the reduction of the acceleration variance, shown by the red bars in Figure 5, is demonstrated. At time $t = 140s$ the leader stands still, and it can be observed, that all following vehicles are keeping the minimum distance $g_{\min}$ to the leader.

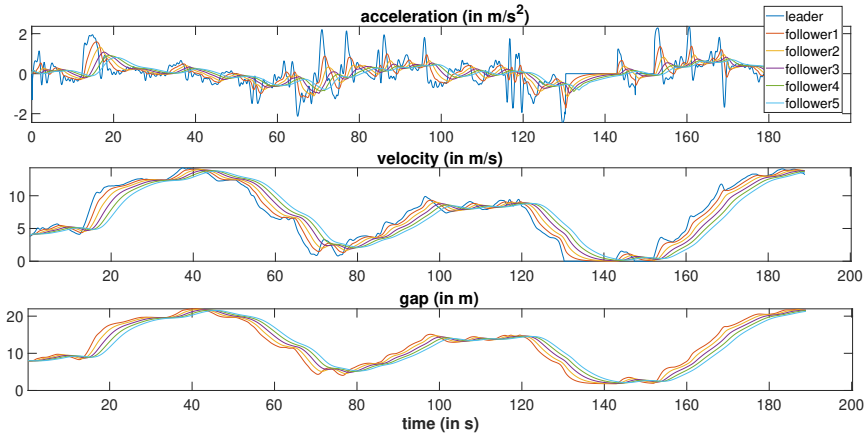

Figure 7: Response to a real leader trajectory

5.4. Response of different driver characteristics

As mentioned in Section 4.2, different driving styles can be achieved by adjusting the parameters of the reward function. Three RL agents has been trained on a reward function, that differs in the desired time gap T between following and leading vehicle ($T_1 = 1.0s$, $T_2 = 1.5s$, $T_3 = 2.0s$). Figure 8 shows the result of these agents, following the real leader trajectory from Napoli. It can be observed, that a lower value for T results in closer driving to the leader which can be considered as a more "aggressive" driving style. Since this also means that there are less options in increasing driving comfort without affecting safety, the follower's accelerations and decelerations are higher than that of the more conservative RL agent 3 although the relative weighting $w_{\text{jerk}}/w_{\text{gap}}$ of the safety and comfort aspects in Eq. (5) and (16) has not been changed. Still, when simulating platoons of the three RL agents in any order, the accelerations and jerks decrease along the platoon. [Probeweise simulieren, obwohl ich sicher bin, dass das der Fall ist. Durchaus auch mal extreme Fälle wie $T = 0.5\,\text{s}$ oder gar $0.2\,\text{s}$ (mit $d = 0.05\,\text{s}$) ausprobieren und schauen, was passiert!]

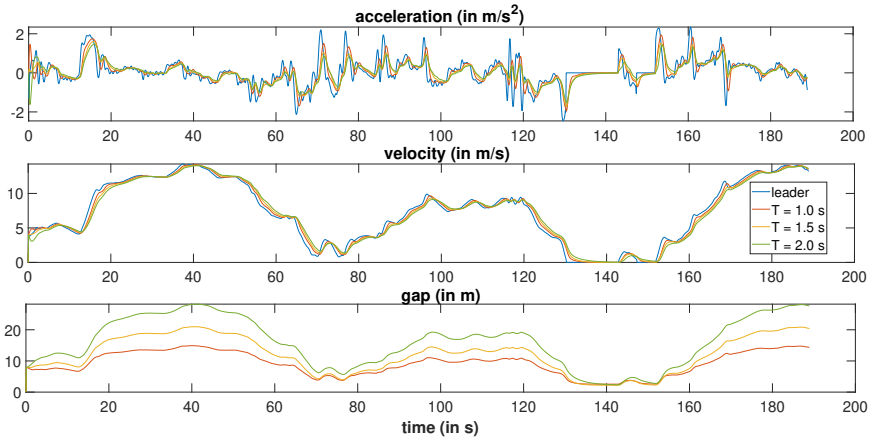

Figure 8: Impact of differently parametrized RL agents on the driving behavior

5.5. Cross validation with the Intelligent Driver Model

To compare the performance of the RL agent with that of classical car-following models, we chose the commonly used Intelligent-Driver Model (IDM)Treiber et al. (2000) whose acceleration is given by [here, I have left s^* since g^* has already been used for the gap reward function]

$$a = a_{\max}\left(1 - \left(\frac{v}{v_{\text{des}}}\right)^4 - \left(\frac{s^*\left(v, v_l - v\right)}{g}\right)^2\right), \tag{26}$$

with

$$s^*\left(v, v_l\right) = g_{\min} + \max\left(0, vT + \frac{v(v - v_l)}{2\sqrt{a_{\max} b_{\text{comf}}}}\right). \tag{27}$$

Notice that the IDM parameters desired speed v_{des}, minimum gap $g_{\min}$, time gap T, maximum acceleration $a_{\max}$, and comfortable deceleration b_{comf} are a subset of that of the RL reward function.

First, we calibrate the IDM on the Napoli data set by minimizing the sum of squares of the relative gap error, SSE($\ln g$), of the first follower with respect to the data (cf. Table 4). The same parameters are also assumed for the reward function of the RL agent before it was trained on the artificial AR(1) generated leader speed profile. Notice that the RL agent used the Napoli data only indirectly by parameterizing its reward function. [Du solltest auch mal testen,

Table 4: IDM parameters calibrated to the Napoli data and also used for the reward function of the RL agent

Parameter	Value
T	$0.83s$
$g_{\min}$	$4.90m$
$a_{\max}$	$4.32m/s^2$
b_{comf}	$2.34m/s^2$
v_{des}	$33.73m/s$

was passiert, wenn du die Reward-Funktionen zum Lernen anders parameterisiert. So ist es ja ein wenig unfair, da der RL-Agend beim Lernen schon die volle IDM-Information hat und durch die Dutzende (Hunderte?) von internen Synapsenparameter der verschiedenen Layers durch mehr Parameter eigentlich nur besser werden kann. Ich gehe davon aus, dass dann die Fitfunktion irgendwann schlechter wird, aber die Reward-Funktion nicht. Just exploring; vielleicht ergeben sich interessante Erkenntnisse ...]

Figure 9 shows the results for: First, the RL agent, calibrated on the real follower data. Second, the IDM, calibrated on the follower data. And third, the real follower of the Napoli experiment. To evaluate the performance, for both approaches the respective objective function has been computed. The objective function of the RL agent corresponds to the reward function, while the Goodness-of-Fit Function $SSE(g)$ defines the objective function of the IDM. A comparison between RL agent and IDM for both, the reward function and the Goodness-of-Fit Function, is shown in Table 5. For both objectives the RL agent shows a better performance.

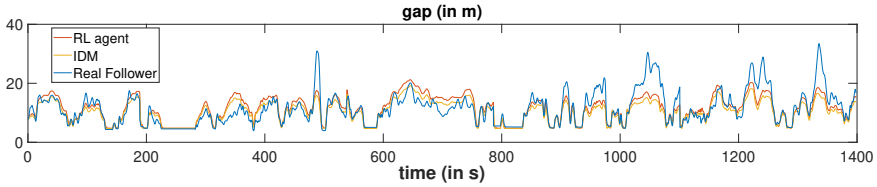

Figure 9: Comparison between IDM and RL agent, calibrated on the same set of parameters

Table 5: Comparison between calibrated RL agent and IDM for accumulated Reward and Goodness-of-Fit Function $SSE(log(g))$

	RL agent	IDM
$SSE(log(g))$	389.10	418.05
Accumulated Reward	6.86×10^3	6.73×10^3

6. Conclusion/Discussion

This study presented a novel car-following model based on DRL. The proposed model considers free driving, car-following, as well as the transition between both in a way, that approaching the leading vehicle is smooth and comfortable. We used an approach called Modularized Reinforcement Learning (Bhat et al. (2006)) to decompose the multi-goal problem into two subtasks. Two different RL policies have been designed using the Deep Deterministic Policy Gradient algorithm. The Free-Driving-Policy refers to free driving and aims not to exceed a desired speed. The Car-Following-Policy refers to following a vehicle and aims to keep a reasonable gap to a leader vehicle.

For each policy, we defined separate reward functions, which reflect traffic safety and comfort aspects. Different driver characteristics can be modeled by adjusting the parameter of the reward function.

The proposed model is trained on leading trajectories based on an AR(1)-process. This leads to high generalization capabilities and a model usage in a wider variety of traffic scenarios. For the evaluation of the trained agents, differ-

ent traffic scenarios have been simulated. The scenarios based on both synthetic and real trajectory data,[Die Napoli-Daten sind *nicht* *naturalistic*. So nennt man nur Daten von Fahrern, die nicht wissen, dass sie beobachtet werden; die Napoli-Daten sind im Gegensatz dazu *experimental* data, aber natuerlich *real*] bringing the model to its limits. In all cases, the car-following model proved to be accident-free and comfortable. Further scenarios showed that traffic oscillations could effectively be dampened with a sequence of trained followers, even if the leader shows large outliers in acceleration.

U. D. of Transportation National Highway Traffic Safety Administration, Critical reasons for crashes investigated in the national motor vehicle crash causation survey, NHTSA, Washington, DC, USA (2015).

P. Wang, C.-Y. Chan, Autonomous ramp merge maneuver based on reinforcement learning with continuous action space (2018).

Y. Lin, J. McPhee, N. Azad, Anti-jerk on-ramp merging using deep reinforcement learning, 2020, pp. 7–14. doi:10.1109/IV47402.2020.9304647.

D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, K. Fujimura, Navigating occluded intersections with autonomous vehicles using deep reinforcement learning, 2018, pp. 2034–2039. doi:10.1109/ICRA.2018.8461233.

Y. Gong, M. Abdel-Aty, J. Yuan, Q. Cai, Multi-objective reinforcement learning approach for improving safety at intersections with adaptive traffic signal control, Accident Analysis & Prevention 144 (2020) 105655.

G. Tolebi, N. S. Dairbekov, D. Kurmankhojayev, R. Mussabayev, Reinforcement learning intersection controller, in: 2018 14th International Conference on Electronics Computer and Computation (ICECCO), 2018, pp. 206–212. doi:10.1109/ICECCO.2018.8634692.

P. Wang, C. Chan, A. de La Fortelle, A reinforcement learning based approach for automated lane change maneuvers, in: 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 1379–1384. doi:10.1109/IVS.2018.8500556.

M. Treiber, A. Hennecke, D. Helbing, Congested traffic states in empirical observations and microscopic simulations, Physical Review E 62 (2000) 1805–1824.

M. Treiber, A. Kesting, The intelligent driver model with stochasticity – new insights into traffic flow oscillations, Transportation Research Part B: Methodological 117 (2018) 613 – 623. TRB:ISTTT-22.

L. Chong, M. M. Abbas, A. Medina, Simulation of driver behavior with agent-based back-propagation neural network, Transportation Research Record 2249 (2011) 44 – 51.

M. Zhou, X. Qu, X. Li, A recurrent neural network based microscopic car following model to predict traffic oscillation, Transportation Research Part C: Emerging Technologies 84 (2017) 245–264.

M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, R. Ke, Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving, Transportation Research Part C: Emerging Technologies 117 (2020) 102662.

M. Zhu, X. Wang, Y. Wang, Human-like autonomous car-following model with deep reinforcement learning, Transportation Research Part C: Emerging Technologies 97 (2018) 348–368.

Y. Lin, J. McPhee, N. Azad, Comparison of deep reinforcement learning and model predictive control for adaptive cruise control (2019).

W. Yuankai, H. Tan, J. Peng, B. Ran, A deep reinforcement learning based car following model for electric vehicle, Smart City Application 2 (2019).

X. Qu, Y. Yu, M. Zhou, C.-T. Lin, X. Wang, Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: a reinforcement learning based approach, Applied Energy 257 (2020) 114030.

A. R. Kreidieh, C. Wu, A. M. Bayen, Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 1475–1480. doi:10.1109/ITSC.2018.8569485.

L. Jiang, Y. Xie, D. Chen, T. Li, N. Evans, Dampen the stop-and-go traffic with connected and automated vehicles- a deep reinforcement learning approach, 2020.

J. Honerkamp, Stochastic dynamical systems: concepts, numerical methods, data analysis, John Wiley & Sons, 1993.

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, CoRR (2015).

S. Bhat, C. Jr, M. Mateas, On the difficulty of modular reinforcement learning for real-world partial programming., volume 1, 2006.

M. Cai, M. Hasanbeig, S. Xiao, A. Abate, Z. Kan, Modular deep reinforcement learning for continuous motion planning with temporal logic, 2021. arXiv:2102.12855.

Z. Wang, Z. Tian, J. Xu, R. K. V. Maeda, H. Li, P. Yang, Z. Wang, L. H. K. Duong, Z. Wang, X. Chen, Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system, in: 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017, pp. 684–689. doi:10.1109/ASPDAC.2017.7858403.

J. Andreas, D. Klein, S. Levine, Modular multitask reinforcement learning with policy sketches (2016).