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

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

1. Introduction

Autonomous driving technologies offer promising solutions to improve road safety, where human errors account for 94% of the total accidents [1]. Moreover, congestion, energy consumption, and emissions are intended to be reduced. However, autonomous driving is a challenging task since traffic flow can be dynamic and unpredictable. On the way to fully autonomous driving, Advanced Driver Assistance Systems (ADAS) have been developed to solve tasks like car following, emergency braking, lane-keeping, or lane changing. Since Deep Learning methods that use deep neural networks have 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 using deep neural networks with Reinforcement Learning, has shown its potential in a wide variety of autonomous driving tasks. In [2] and [3], DRL is used to guide an autonomous vehicle safely via an on-ramp to the freeway. In [4], [5] and [6], DRL methods are used to manage traffic of

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

autonomous vehicles at intersections, optimizing safety and efficiency. In [7], DRL is used to solve lane-changing maneuvers.

A further task in autonomous driving is to model the vehicle behavior under car-following scenarios, where suitable accelerations must be computed to achieve safe and comfortable driving. Approaches for solving this task are classical car-following models, such as the IDM [8] or stochastic car-following models such as that of [9]. Furthermore, data-driven approaches used Deep Learning methods to train a car-following model based on experimental car-follower data, such as in [10] or [11]. The downside of this approach is that the model tries to emulate human driver behavior, which can still 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 [12] or [13]. Similar approaches suggest a standardized driving cycle serving as a leading vehicle trajectory, such as [14] or [15] who use the New European Driving Cycle. A disadvantage coming along with these approaches is that the performance decreases for scenarios that are not in the scope of the training data, indicating inadequate machine learning generalization [14].

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 these by [16], [17], and [18].

All the mentioned DRL car-following models have three disadvantages in common: First, the acceleration range is limited in a way that full-braking maneuvers 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 proven.

This motivated us to design a model that overcomes all these issues. To our

knowledge, no RL based car-following model has been proposed that has the following features combined:

- free driving, car-following, as well as the transition between both in a the way that approaching the leading vehicle is always smooth and comfortable
- wide range of possible accelerations leading to a collision-free behavior also in safety-critical situations such as a full-braking maneuver of the leader
- being trained on leading trajectories, based on an Ornstein–Uhlenbeck process [19] with parameters reflecting the kinematics of real drivers. This leads to high generalization capabilities and model usage in a wider variety of traffic scenarios. Moreover, the supply of learning data is unlimited.
- different driver characteristics to be modeled by adjusting the parameters of the reward function.
- string stability even in extreme situations

Another feature of this work is a thorough validation of the above-mentioned properties in scenarios based on both synthetic and real 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 the IDM calibrated on the same data. The results indicate a better performance of the proposed model. This work is structured as follows. In Section 2, we introduce some basic RL background as well as our modularized RL approach, followed by a detailed description of our two sub-modules: the Free-Driving Policy in Section 3 and the Car-Following Policy in Section 4. In Section 5, we evaluate our RL strategy with respect to safety, *stability*, and comfort aspects before we conclude in Section 6.

2. Reinforcement Learning Background

The follower 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 deterministic and continuous action and a continuous state space, such as in [12], [14] or [13]. The original work can be found in [20]. DDPG is an actor-critic method, that uses an actor network $\mu\left(s \mid \theta^{\mu}\right)$ with the network weights θ^{μ} to output an action a based on a given state s and a critic network $Q\left(s, a \mid \theta^{Q}\right)$ with the network 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 the actor and critic networks. These target networks have the same architecture as the main networks, only differing in the network parameters. While training, these networks are updated by letting them slowly track the main networks

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \tag{2}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}, \tag{3}$$

where τ defines the soft target update rate. This approach greatly enhances the stability of learning. 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 autoregressive process with Gaussian innovations. The entire DDPG algorithm is shown in Algorithm 1.

Algorithm 1: DDPG algorithm according to [20]

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

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

  - 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 decom-

position 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 ([21]). Numerous works are using a modularized Reinforcement Learning approach, like in [22], [23] or [24] just to name 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. We adopted this approach from the IDM+ that also uses separate terms for free-flow and interaction with a leading vehicle ([25]). In the next sections, the model specifications of the Free-Driving Policy and the Car-Following Policy, 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. In our use case, the action space is continuous and one-dimentional and given by the range of feasible accelerations $\dot{v}_t \in [a_{\min}, a_{\max}]$. The range limits $a_{\min} = -9\,\mathrm{m/s^2}$ and $a_{\max} = 2\,\mathrm{m/s^2}$ reflect comfortable driving

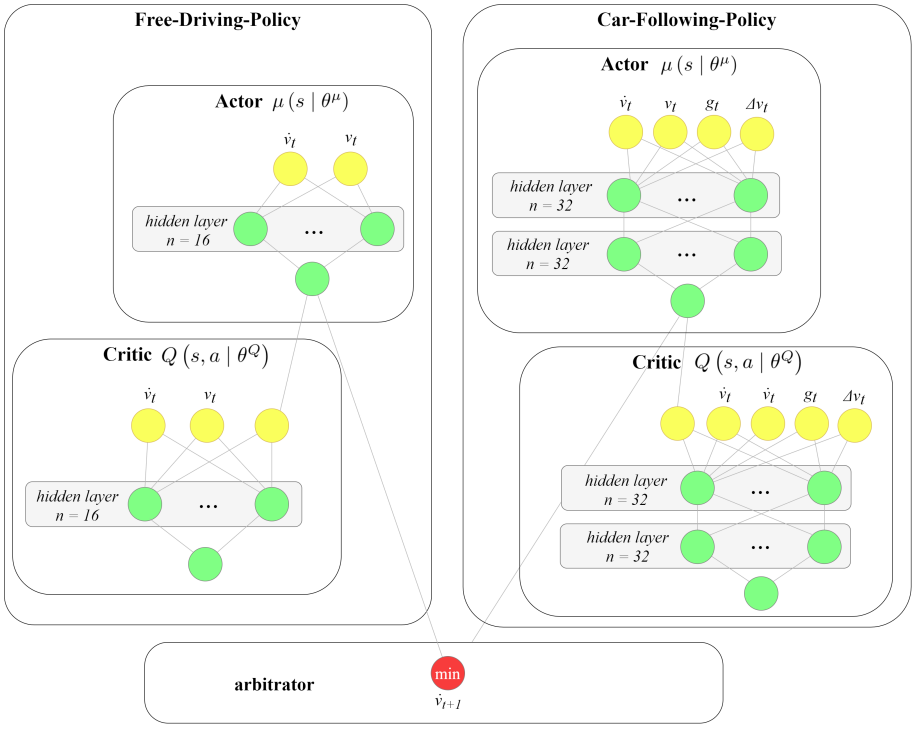

Figure 1: Modularized RL architecture with actor networks of both policies.

($\dot{v}_t \leq a_{\max}$) while allowing for the physical limit $-a_{\min}$ of the braking deceleration in safety-critical situations.

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 $\dot{v}$, and its own speed v. Linear translation and scaling are used to reduce the dimensions and to bring all observations approximately into the same range of $[0, 1]$. The observation vector at time step t is defined as

$$s_t = \begin{pmatrix} \frac{v_t}{v_{\text{des}}} \\ \frac{\dot{v}_t - a_{\min}}{a_{\max} - a_{\min}} \end{pmatrix}, \tag{4}$$

where v_{des} defines the desired speed.

3.2. Reward Function

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

$$r_{t,1} = \begin{cases} \frac{v_t}{v_{\text{des}}}, & \text{if } v_t < v_{\text{des}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

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

$$r_{t,2} = -\left(\frac{1}{j_{\text{comf}}} \, \frac{\mathrm{d}\dot{v}_t}{\mathrm{d}t} \right)^2, \tag{6}$$

where j_{comf} denotes a comfortable jerk of $2\,\mathrm{m/s^3}$. Since building up a comfortable value of acceleration or deceleration from zero in one second is at the limit of comfortable jerks, $r_{t,2}$ 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

$$r_t = r_{t,1} + w_{\text{jerk}} r_{t,2}, \tag{7}$$

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. For the description of the vehicle dynamics, a point-mass kinematic model is used. The Euler and ballistic methods are used to update the speed and position for time step $t+1$. This approach is recommended in [26] as an efficient and robust scheme for integrating car-following models.

$$v_{t+1} = v_t + \dot{v}_t d, \tag{8}$$

$$x_{t+1} = x_t + \frac{v_t + v_{t+1}}{2}\Delta t, \tag{9}$$

with Δt corresponding to the simulation step size that is globally set to 100 ms. To train the RL agent, a training episode has to be defined. One training episode contains 500 time steps, and the vehicle's initial speed is set uniformly at random in the range $[0, v_{\text{des}}]$.

3.4. Model training

Both neural networks, critic and actor, are feed-forward neural networks with one layer of hidden neurons, containing 16 neurons (see Figure 1). ReLU activation functions ([27]) are used, except for the output layer of the actor network that uses a tanh activation function. The bounded output in the range $[-1, 1]$ is then mapped in the acceleration range. The learning rate for updating the weights of the critic and actor network 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 as suggested in [20]

$$dx_t = -\theta x_t dt + \sigma dW_t, \tag{10}$$

with $\Theta = 0.15$, $\sigma = 0.2$ and W_t denoting the Wiener process.

The soft update rate of the target networks τ is set to 0.001. All DDPG parameters are presented in Table 1. Figure 2 shows an example of the training

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

process, where the asymmetric moving average reward of the last 30 training episodes is plotted over training episodes. For the Free-Driving Policy, a maximum average reward has been reached after approximately 3200 training episodes (marked in red).

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 $a_{\min} = -9\,\mathrm{m/s^2}$ and $a_{\max} = 2\,\mathrm{m/s^2}$.

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 $\dot{v}_t$, its own speed v_t, the leader's speed $v_{t,l}$, and the (bumper-to-bumper) gap g_t. Again, linear translation and scaling are used to reduce the dimensions and to bring all observations approximately into the same

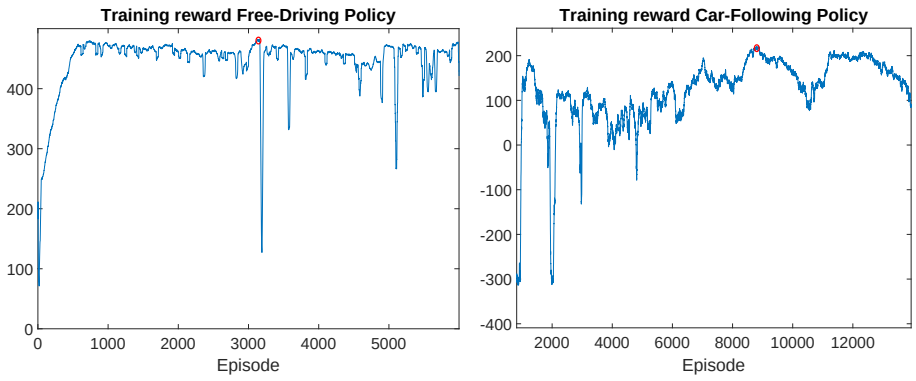

Figure 2: Asymmetric moving average reward of the last 30 training episodes for Free-Driving Policy and Car-Following Policy

range of $[-1, 1]$. The observation at time step t is defined as

$$s_t = \begin{pmatrix} \frac{v_t}{v_{\text{des}}} \\ \frac{\dot{v}_t - a_{\min}}{a_{\max} - a_{\min}} \\ \frac{v_{t,l} - v_t}{v_{\text{des}}} \\ \frac{g_t}{g_{\max}} \end{pmatrix}, \tag{11}$$

where $g_{\max}$ is set to 200 m. When g_t exceeds $g_{\max}$ or there is no leader, g is set to $g_{\max}$.

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

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	$2\,\mathrm{m/s^2}$
j_{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 Equation (5))	$15\,\mathrm{s}$
w_{gap}	relative weight for keeping the desired gap	0.5
w_{jerk}	relative weight for comfortable acceleration	0.004

of the leader) with the comfortable deceleration b_{comf},

$$r_{t,1} = \begin{cases} -\tanh\left(\frac{b_{kin}-b_{\mathrm{comf}}}{-a_{\min}}\right), & \text{if } b_{kin} > b_{\mathrm{comf}} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

with

$$b_{kin} = \begin{cases} \frac{(v_t - v_{t,l})^2}{g_t}, & \text{if } v_t > v_{t,l} \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

the kinematic deceleration b_{kin} representing the minimum deceleration necessary to avoid a collision. 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 (12) ensures that this term is not activated in non-critical situations. The purpose of the factor $r_{t,1}$ is twofold: It

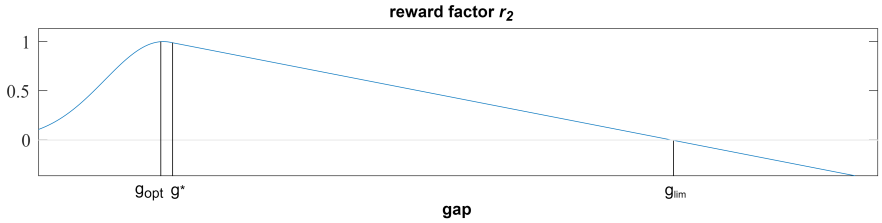

Figure 3: Factor 2 of the reward function maximizing the reward for car following with time gap T

motivates the follower vehicle to brake in safety-critical or near-crash situations. Furthermore it motivates the follower vehicle also to brake early in non-critical situations, where $v_t > v_{t,l}$, in order to achieve a comfortable approaching of the leader vehicle.

The second part of the reward function aims to not fall below a reasonable distance to the leading vehicle.

$$r_{t,2} = \begin{cases} \frac{\varphi\{(g_t - g_{\text{opt}})/g_{\text{var}}\}}{\varphi(0)}, & \text{if } g_t < g^* \\ \frac{\varphi\{(g_t - g_{\text{opt}})/g_{\text{var}}\}}{\varphi(0)} \left(1 - \frac{g_t - g^*}{g_{\text{lim}} - g^*}\right) & \text{otherwise} \end{cases} \tag{14}$$

with

$$g_{\text{opt}} = v_t T + g_{\text{min}}, \tag{15}$$

$$g_{\text{var}} = 0.5 g_{\text{opt}}, \tag{16}$$

$$g_{\text{lim}} = v_t T_{\text{lim}} + 2 g_{\text{min}}, \tag{17}$$

and $\varphi(x)$ describing the density function of the standard normal distribution. The value of g^* is chosen in a way that the reward function $r_{t,2}$ is differentiable. Figure 3 illustrates the reward function for $r_{t,2}$, 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 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 $g_t > g^*$ have been applied, but the best results regarding a smooth and comfortable approaching of the following vehicle have been reached with a linear function. Furthermore, a high value of T_{lim} has been chosen, resulting in a low gradient of the linear function. This prevents the follower vehicle from trying to reach the desired time gap T as fast as possible with maximum speed but rather motivates the follower vehicle to decelerate early and approach the leading vehicle comfortably.

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,

$$r_{t,3} = -\left(\frac{1}{j_{\mathrm{comf}}}\ \frac{\mathrm{d}\dot{v}_t}{\mathrm{d}t}\right)^2. \tag{18}$$

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

$$r_t = r_{t,1} + w_{\mathrm{gap}} r_{t,2} + w_{\mathrm{jerk}} r_{t,3}, \tag{19}$$

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

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 RL agent controls the follower vehicle, the leading trajectory is based on an Ornstein–Uhlenbeck process [19], whose parameters reflect the kinematics of real leaders. The Ornstein–Uhlenbeck process describes the speed of the leading vehicle $v_{t,l}$ and is defined as

$$\mathrm{d}v_{t,l} = \theta\left(\mu - v_{t,l}\right)\mathrm{d}t + \sigma \mathrm{d}W_t, \tag{20}$$

with $\theta = 0.132\,\mathrm{s}^{-1}$, $\mu = 7.5\,\mathrm{m/s}$ and $\sigma = 3.847\,\mathrm{m/s}^{1.5}$. W_t denotes the Wiener process. The parameter have been designed to model the behavior of a real

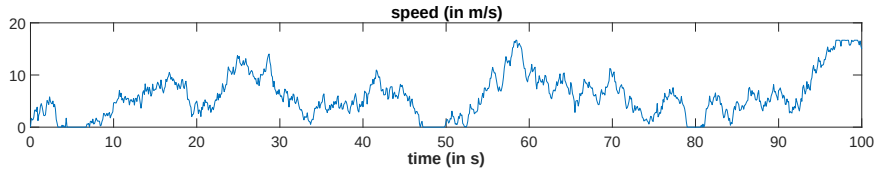

Figure 4: Example of a leading trajectory based on the parametrized Ornstein-Uhlenbeck process used to train the RL agent

leader vehicle. For the numerical simulation, the Ornstein-Uhlenbeck process has been discretized using the Euler-Maruyama method. The resulting discretized process is defined as

$$v_l(n+1) = v_l(n) + \theta\left(\mu - v_l(n)\right)\Delta t + \sigma \Delta W_n, \tag{21}$$

with

$$\Delta W_n \sim N(0, \Delta t) \tag{22}$$

Figure 4 shows an example trajectory of the leading vehicle. After the Ornstein-Uhlenbeck process is calculated for one episode, all speed values are clipped to the range $[0, 16.6\,]m/s$. This generates training situations, where the leader vehicle stands still for some time, e. g. at $t = [48\,\mathrm{s}, 50\,\mathrm{s}]$ in Figure 4. Furthermore, the leader's trajectory also contains intervals where the speed is above the desired speed $v_{\mathrm{des}} = 15\,\mathrm{m/s}$ of both the leader and the follower, e.g., around $t = 58\,\mathrm{s}$ and $t \in [97\,\mathrm{s}, 100\,\mathrm{s}]$. One training episode has a simulation time of 50 s. With a step size of $\Delta t = 100\,\mathrm{ms}$, this results in an episode length of 500 steps. The initial speeds of the following and leading vehicles are both set uniformly at random in the range $[0, v_{\mathrm{des}}]$, respectively. The initial space gap g_0 between both is set to 120 m.

4.4. Model training

Both neural networks, critic and actor, are feed-forward neural networks with two layers of hidden neurons, containing 32 neurons each (see Figure 1). Except for this configuration, the architecture of the critic and actor, as well as the

parameter of the DDPG algorithm, is the same as we used for the Free-Driving Policy (see Table 1).

Figure 2 shows an example of the training process, where the asymmetric moving average reward of the last 30 training episodes are plotted over training episodes. For the Car-Following Policy, a maximum average reward has been reached after approximately 8900 training episodes (marked in red). The training processes for both policies are quite unstable. As this is a known issue using the DDPG algorithm, we plan to use a more stable algorithm like the TD3 algorithm [28].

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 to evaluate the transition between free driving and car-following as well as the follower's behavior in safety-critical situations. Figure 5 shows a driving scenario with an artificial external profile for the leading vehicle speed. The initial gap between follower and leader is 200 meters referring to a free driving scenario. The follower accelerates with $a_{\max} = 2\,\mathrm{m/s^2}$ until the desired speed $v_{\mathrm{des}} = 15\,\mathrm{m/s}$ is reached and approaches the standing leading vehicle. When the gap between both drops below 70 m, the follower starts to decelerate with a maximum deceleration of approximately $b_{\mathrm{comf}} = 2\,\mathrm{m/s^2}$ (transition between free driving and car-following) and comes to a standstill with a final gap of approximately $g_{\min} = 2\,\mathrm{m}$. Afterwards ($t = 30\,\mathrm{s}$), the leading vehicle accelerates to a speed that is below the desired speed of the follower before performing a maximum braking maneuver ($\dot{v}_l = -9\,\mathrm{m/s^2}$) to a full stop ($t = 46\,\mathrm{s}$). At the time of the start of the emergency braking, the follower has

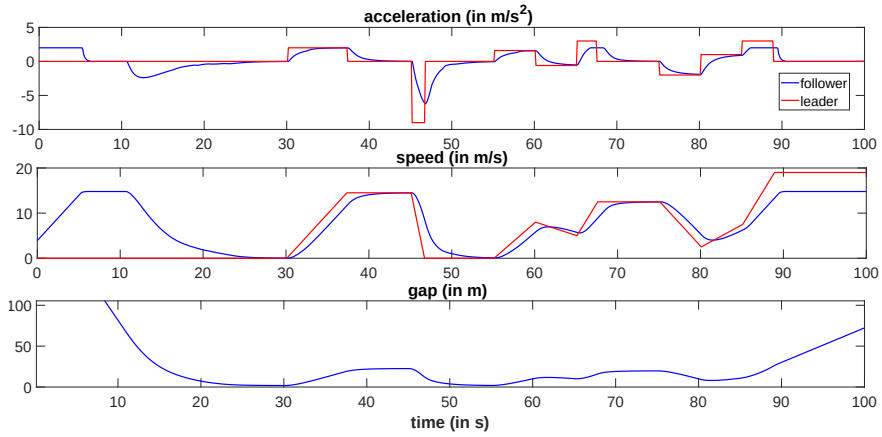

Figure 5: Response to an external leading vehicle speed profile.

nearly reached a steady following mode at the desired space gap $g_t = g_{\min} + v_t T$. 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 $-\dot{v} \approx 5\,\mathrm{m/s^2}$. The transition between different accelerations happens in a comfortable way reducing the resulting jerk. Only at the beginning ($t = 46\,\mathrm{s}$) where the situation is really critical, the jerk $\mathrm{d}\dot{v}/\mathrm{d}t$ 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 $g_{\min} + v_t T$ 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 7, consists of a leader based on the Ornstein-Uhlenbeck process from Section 4.3 that is followed by five vehicles, each controlled by the trained RL agent. The results show that traffic oscillations can effectively be dampened with a sequence of trained followers,

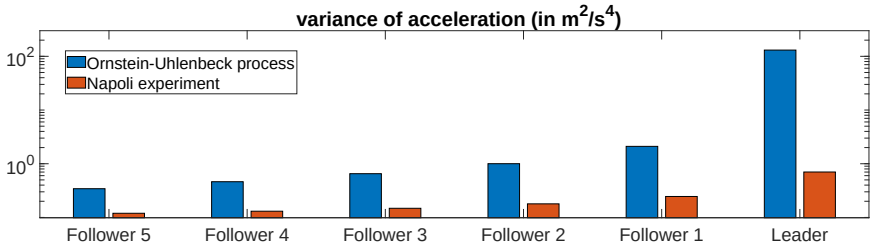

Figure 6: Comparison of the acceleration variance between leader and follower for a leader controlled by the Ornstein-Uhlenbeck process (blue bars) and the leading vehicle of the Napoli experiment (red).

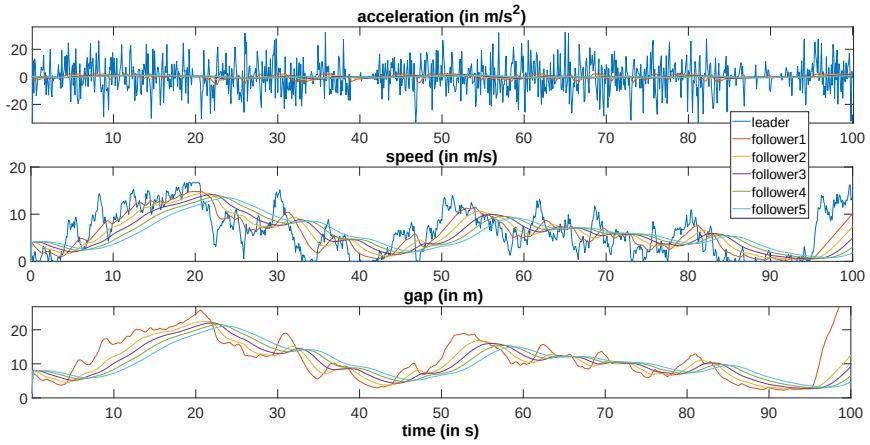

Figure 7: Response to a leader trajectory based on a Ornstein-Uhlenbeck process

even if the leader shows large outliers in acceleration. Figure 6 illustrates the difference of accelerations between the leader and the followers (blue bars). The last follower shows the lowest variance of acceleration demonstrating 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 (Figure 8). This trajectory comes from the platoon driving experiments of [29] on urban and peripheral arterials in Napoli where high-precision distance data between the platoon vehicles were obtained. Similar

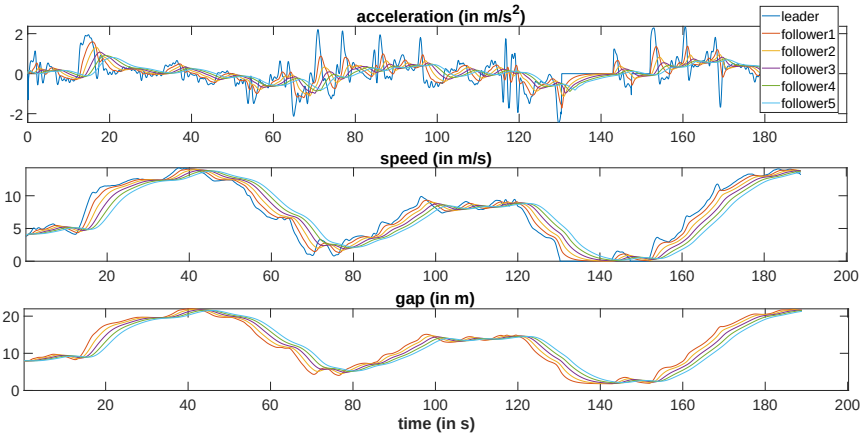

Figure 8: Response to a real leader trajectory

to the experiment from Section 5.2, string stability and the reduction of the acceleration variance, shown by the red bars in Figure 6, is demonstrated. At time $t = 140\,\mathrm{s}$ the leader stands still, and it can be observed that all following vehicles are keeping the minimum distance $g_{\min}$ to the leader.

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 9 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. The last plot in Figure 9 which illustrates the actual time gap $(g_t - g_{\min})/v_t$ shows, that for speeds greater than zero the desired time gap T is approximately maintained. This behavior can be considered as a more "aggressive" driving style. Since this also means that there are fewer 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 $w_{\mathrm{jerk}}/w_{\mathrm{gap}}$ of the safety and comfort aspects in Equation (7) and (19)

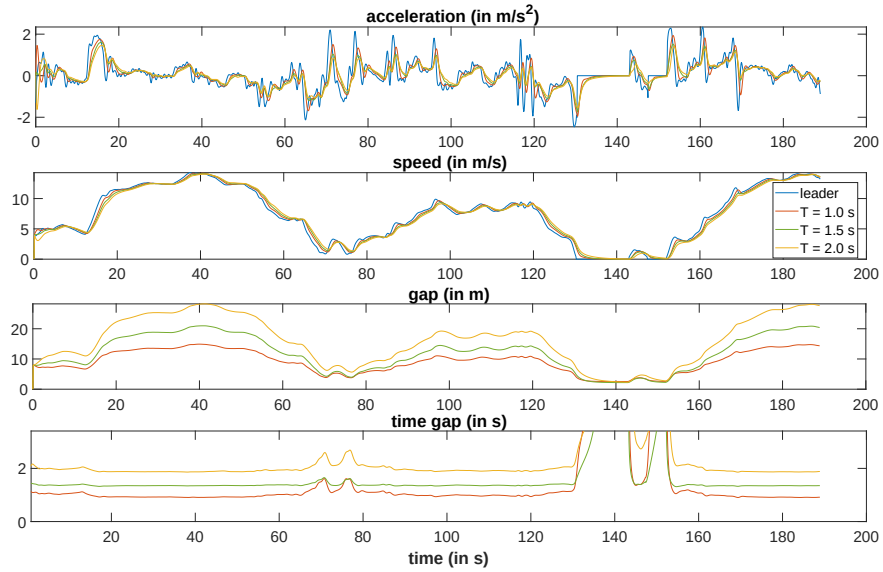

Figure 9: Impact of differently parametrized RL agents on the driving behavior

has not been changed. Still, when simulating platoons of the three RL agents in any order, the accelerations and jerks decrease along the platoon.

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 IDM of [8] whose acceleration is given by

$$\frac{\mathrm{d}v}{\mathrm{d}t} = a_{\max}\left[1 - \left(\frac{v_t}{v_{\mathrm{des}}}\right)^4 - \left\{\frac{s^*\left(v_t, v_{t,l}\right)}{g_t}\right\}^2\right], \tag{23}$$

with

$$s^*\left(v_t, v_{t,l}\right) = g_{\min} + \max\left\{0, v_t T + \frac{v_t(v_t - v_{t,l})}{2\sqrt{a_{\max} b_{\mathrm{comf}}}}\right\}. \tag{24}$$

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

Table 3: IDM parameters calibrated to the Napoli data and also used for the reward function of the RL agent

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

to the data (cf. Table 3). The same parameters are also assumed for the reward function of the RL agent before it was trained on the artificial Ornstein-Uhlenbeck generated leader speed profile. Notice that the RL agent used the Napoli data only indirectly by parameterizing its reward function.

Figure 10 shows the results for (i) the RL agent, calibrated on the real follower data (red lines), (ii) the IDM, calibrated on the same follower data (amber), and (iii) the real follower of the Napoli experiment (red). To compare the performance for both approaches, the respective objective functions have been computed. The objective function of the RL agent corresponds to the reward function, while the goodness-of-fit function SSE($\ln g$) defines the objective function of the IDM. Furthermore, we cross-compared the values by calculating the reward function for the IDM and SSE($\ln g$) for the RL model. All values are shown in Table 4. As expected, the RL agent performs better than the IDM relative to the reward function used for its learning. It is remarkable, however, that the RL model also outperforms the IDM relative to the the goodness-of-fit function, that has been used only indirectly by parameterizing its reward function.

5.6. Time-to-collision comparison with the IDM

An important safety measure for car-following models is the time-to-collision (TTC). Again we took leading trajectories coming from platoon driving exper-

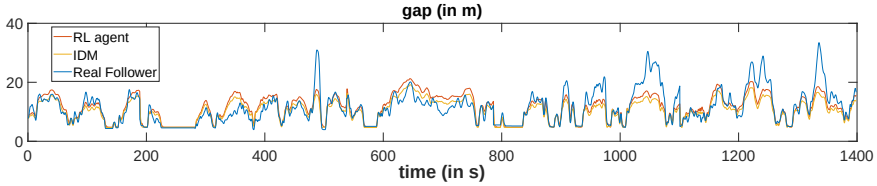

Figure 10: Comparison between IDM and RL agent, calibrated on the same set of parameters

Table 4: Comparison between calibrated RL agent and IDM for accumulated Reward and Goodness-of-Fit Function SSE($\ln g$)

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

iments in Napoli to conduct a TTC analysis. We simulated 15 different leaders from those experiments, once followed by an RL agent and the other time by an IDM vehicle (Sect. 5.5). Both followers are parametrized according to the default parameters of Table 2. Figure 11 shows the distribution of TTC for all 15 experiments. The TTC has been evaluated at each simulation time step. The lower TTC bound for the RL agent is 1.99 s, while the one for the IDM vehicle is 2.04 s. But except for some values around TTC = 2 s, the TTC values of the RL agent show to be higher than those of the IDM.

The difference in driving behavior is further evaluated by focusing on a single leader-follower experiment, shown in Figure 12, where the situation with the lowest TTC for the RL agent is marked green. In this situation, the leader brakes to a standstill with a quite high deceleration of $-\dot{v}_l \approx 6\,\mathrm{m/s^2}$. The reason for the slight difference in TTC between IDM and RL agent lies in the fact, that the IDM keeps a slightly higher gap to the leader most of the time, which simply reflects a more defensive driving style.

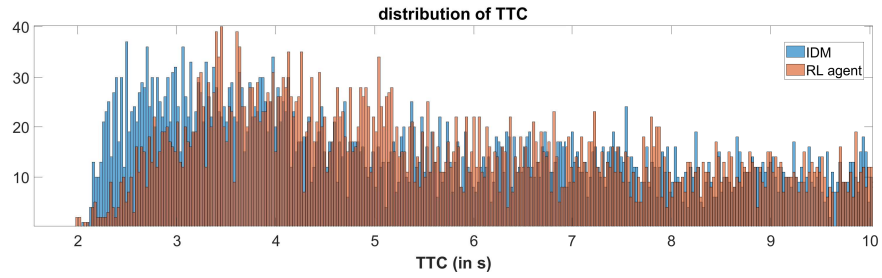

Figure 11: Distribution of time-to-collision (TTC) for IDM and RL agent in different experiments based on real leader trajectories

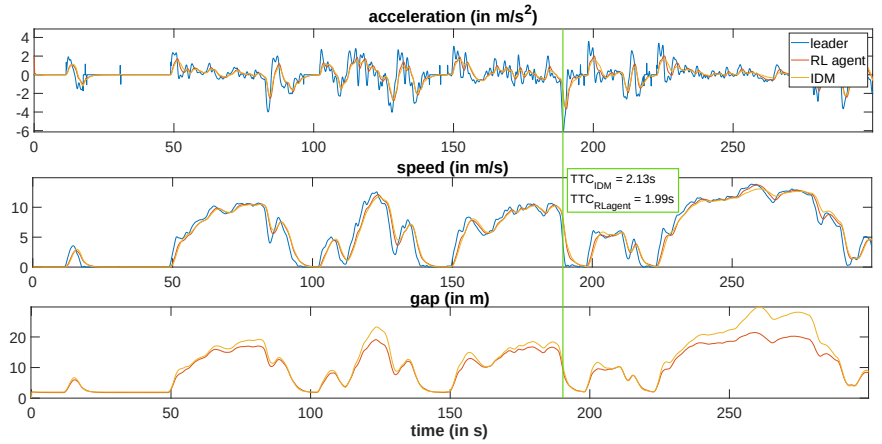

Figure 12: Response of an IDM and RL agent on a real trajectory. Safety-critical situations where the TTC of the IDM drops below 2.5 s are marked in green.

6. Conclusion/Discussion

This study presented a novel car-following model based on Deep Reinforcement Learning. 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 ([21]) 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 aims to reach and not exceed a certain desired speed. The Car-Following Policy aims to keep a reasonable gap to a leader vehicle and keep the traffic situation safe.

For each policy, we defined separate reward functions reflecting 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 Ornstein-Uhlenbeck process. This leads to high generalization capabilities and model usage in a wider variety of traffic scenarios. Furthermore, the supply of learning data is unlimited. For the evaluation of the trained agents, different traffic scenarios based on both synthetic and real trajectory data have been simulated, including situations that bring 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.

Besides driving comfort, string stability, and safety, the efficiency of the resulting traffic flow is important, particularly, the fundamental diagram, the maximum flow through a bottleneck that can be sustained and the outflow from a region of congested traffic. Furthermore, we have idealized the state dynamics by assuming that a vehicle can instantaneously take on the acceleration prescribed by the actor while the real control path is more complicated and non-negligible response times happen, particularly for positive accelerations. We expect that RL techniques play out their strengths in such more complex

state dynamics. All this will be investigated in a forthcoming paper.

Acknowledgements

We thank Vincenzo Punzo for making available to us the experimental car-following data used in this paper. This work was funded by ...

References

[1] S. Singh, Critical reasons for crashes investigated in the national motor vehicle crash causation survey, NHTSA, Washington, DC, USA (2015).

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] G. E. Uhlenbeck, L. S. Ornstein, On the theory of the brownian motion, Phys. Rev. 36 (1930) 823–841.

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

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

[22] 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.

[23] 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.

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

[25] W. Schakel, B. Arem, B. Netten, Effects of cooperative adaptive cruise control on traffic flow stability, 2010, pp. 759 – 764. doi:10.1109/ITSC.2010.5625133.

[26] M. Treiber, V. Kanagaraj, Comparing numerical integration schemes for time-continuous car-following models, Physica A: Statistical Mechanics and its Applications 419 (2015) 183–195.

[27] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, ICML'10, Omnipress, Madison, WI, USA, 2010, p. 807–814.

[28] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods (2018).

[29] V. Punzo, D. J. Formisano, V. Torrieri, Nonstationary kalman filter for estimation of accurate and consistent car-following data, Transportation research record 1934 (2005) 2–12.