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

**Fabian Hart** fabian.hart@tu-dresden.de
*Chair of Econometrics and Statistics, esp. in the Transport Sector*
*Technische Universität Dresden*
*01062 Dresden, Germany*

**Ostap Okhrin**
*Chair of Econometrics and Statistics, esp. in the Transport Sector*
*Technische Universität Dresden*
*01062 Dresden, Germany*

**Martin Treiber**
*Chair of Econometrics and Statistics, esp. in the Transport Sector*
*Technische Universität Dresden*
*01062 Dresden, Germany*

## Abstract

We propose and validate a complete car-following model based on deep reinforcement learning that is designed to cover a variety of different free-flow and car-following events with a special focus on safety-critical traffic situations. To increase the coverage of possible car-following events in training, we developed a novel training environment based on stochastic processes. The model's application is further extended to free-driving states with a safe and smooth transition between free-driving and car-following resulting in a complete model, i.e., applicable to any single-lane situation. Four different objectives, particularly safety, efficiency, comfort, and adherence to a desired speed, are defined to model a reward function. Adjusting the parameters of this reward function allows for implementing different driving styles. To avoid the known issue of multi-objective reinforcement learning where too many objectives at once lead to degradation in performance, we use a synchronized modular architecture with two different policies.

To evaluate the robustness of the trained model, we simulate validation scenarios for various parameterizations of the reward functions and for a wide variety of artificial and real leader data that are not covered in training and, moreover, are qualitatively different from the learning data. In all simulated scenarios, the model proved to be accident-free, comfortable, and string-stable, even in extreme situations such as full-braking of the leader vehicle.

**Keywords:** reinforcement learning, car-following model, generalization capabilities, string stability

## 1. Introduction

A crucial 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 Intelligent Driver Model (IDM) (Treiber et al., 2000) or stochastic car-following models such as that of Treiber and Kesting (2018). 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. There are different data-driven approaches using deep learning methods, including supervised learning, to train a car-following model, such as in Chong et al. (2011), Zhou et al. (2017), and Zhang et al. (2018). However, the downside of these approaches is that, by using naturalistic data, such as the simulator NGSIM, the model tries to emulate human driver behavior, which can still be sub-optimal.

To overcome this issue, reinforcement learning (RL) methods train models directly, optimizing metrics such as safety and comfort instead of imitating human driving behavior. Using RL, these models are trained through interaction with a simulation environment with the advantage that they do not require a huge amount of external data. Especially in combination with deep neural networks, RL has already shown its potential in a wide variety of autonomous driving

tasks. In Wang and Chan (2018) and Lin et al. (2020b), RL is used to guide an autonomous vehicle safely via an on-ramp to the freeway. Another approach is to manage traffic of autonomous vehicles at intersections, optimizing safety and efficiency such as in Isele et al. (2018), Gong et al. (2020) and Tolebi et al. (2018). Wang et al. (2018) proved the power of RL to solve lane-changing maneuvers.

There are a few studies that also investigate the application of RL to train car-following models, such as in Zhu et al. (2018), Gao et al. (2019), Yuankai et al. (2019), Zhu et al. (2020), Yen et al. (2020), or Lin et al. (2020a). Since in RL the model learns its behavior in a simulated training environment, the design of the training procedure is crucial. However, all reviewed RL-based car-following studies model the leading vehicle's trajectories in a simulation environment based on real driving data, e.g., NGSIM, standard driving cycles, e.g., the New European Driving Cycle, or even constant speeds. The problem coming along with these approaches is that such 'usual' driving data is lacking in safety-critical situations, such as full-braking of the leader vehicle. This results in a training environment that covers just a small part of the distribution of possible car-following states. Consequently, the trained models are suitable for just a particular type of car-following events being in the scope of the training data, but they cannot be generalized for all kinds of car-following scenarios. Lin et al. (2020a) already emphasized in their RL-based car-following study that there is a significant degradation in performance if the testing scenario falls outside the training data range. All other reviewed studies don't even test the trained agent's robustness in scenarios that have not been seen in training. A further issue in existing RL-based car-following studies is that they do not consider free-driving and the transition to car-following.

To our knowledge, there is no existing RL-based car-following model that (i) covers all kind of car-following events, especially safety-critical events or approaching slow or standing vehicles, and (ii) considers free-driving, car-following, and the smooth transition between the two. Such kind of car-following model is referred to as '*complete*' (Treiber and Kesting, 2013). To close this gap, we propose the first complete RL-based car-following model which is characterized by the following features:

- applicable for all kinds of car-following events and robust in extreme situations by using a training environment based on stochastic Uhlenbeck and Ornstein (1930) processes
- applicable for free-driving and car-following situations with safe, smooth, and comfortable transitions between these states by using a synchronized modular RL architecture

Another issue of car-following models is string stability which means that any changes in speed of an individual vehicle in a platoon of vehicles do not amplify when they propagate upstream through the platoon. There are several RL-based studies focusing on dampening traffic oscillations by using a sequence of trained vehicles, such as these by Qu et al. (2020), Kreidieh et al. (2018), and Jiang et al. (2021). Xu et al. (2021) developed a car-following model to solve 'phantom' traffic jams by obtaining information of multiple vehicles ahead. Since such a behavior is highly desirable, we also aim to evaluate the string stability of the trained car-following in this study.

To prove the trained model's robustness, we perform a thorough validation in scenarios that have not been seen in training. We use real trajectory data to evaluate the performance on usual driving situations, as well as synthetic trajectory data, simulating safety-critical situations to bring the model to its limits. Even in extreme cases, the model proved to be accident-free, comfortable, string stable, and showed a smooth transition from free-driving to car-following.

## 2. Model formulation

### 2.1 RL algorithm

The follower vehicle is controlled by an 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 $\pi$ that maximizes the expected accumulated discounted reward

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

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

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 Zhu et al. (2020), Lin et al. (2020a) or Zhu et al. (2018). The original work can be found in Lillicrap et al. (2015). In our preliminary analysis, DDPG also showed a strong performance relative to a variety of other algorithms, such as the PPO algorithm (Schulman et al., 2017). Therefore, in order to keep the focus of the paper on the RL-based car-following model and not on the comparison of various RL algorithms, we concentrate only on the DDPG throughout this work. DDPG is an actor-critic method, that uses an actor network $\mu\left(s \mid \theta^{\mu}\right)$ with the network weights $\theta^{\mu}$ 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 $\theta^{Q}$ to predict if the action is good or bad, based on a given state $s$ and action $a$. To achieve better training stability, DDPG uses target networks $\mu'$ 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. This approach greatly enhances the stability of learning. Furthermore DDPG uses Experience Replay, that implements a Replay Buffer $B$, where a list of tuples $(s_i, a_i, r_i, s_{i+1})$ for $i \quad t$ 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.

### 2.2 Modular reinforcement learning

Furthermore, we use a modular system architecture to decompose the task of car-following into two parallel subtasks. Modular RL refers to the decomposition of a complex, multi-objective problem into a collection of simultaneously running single objective learning processes, typically modeled as Markov Decision Processes. This approach overcomes the known issue of multi-objective RL (Hayes et al., 2022) where too many objectives have to be solved at once. Typically, these sub-agents 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)). Numerous works are using a modular RL approach, like in Cai et al. (2021), Wang et al. (2017) or Andreas et al. (2017) just to name a few. The advantage of decomposing multiple-objective reward functions with MRL, we also want to use in this work. We divide our car-following problem into two sub-tasks, 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 at each time step. To emphasize the parallel computation, we refer to this approach as synchronized modular RL. A illustration of this architecture is depicted in Figure 7 in the appendix. 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, see Schakel et al. (2010).

### 2.3 Hyperparameters

All neural networks, critics and actors, are feed-forward neural networks. The Free-Driving Policy uses one hidden layer with 16 neurons, and the Car-Following Policy uses two hidden layers with 32 neurons each. ReLU activation functions (Nair and Hinton (2010)) are used, except for the output layer of the actor networks that use $\tanh(\,)$ activation functions.

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 Lillicrap et al. (2015)

$$\mathrm{d}x_t = \quad \theta_1 x_t \mathrm{d}t + \sigma_1 \mathrm{d}W_t, \tag{2}$$

with $\theta_1 = 0.15\,\mathrm{s}^{\ 1}$, $\sigma_1 = 0.2\,\mathrm{s}^{\ 0.5}$ and $W_t$ denoting the Wiener process. The soft update rate of the target networks $\tau$ is set to 0.001. All DDPG parameters are presented in Table 1 in the appendix.

### 2.4 Action and state space

The defined synchronized modular RL approach requires that the action space of both sub-policies are identical. In our use case, the action space is continuous and one-dimensional and given by the range of feasible accelerations $\dot{v}_t \in [\dot{v}_{\mathsf{min}}, \dot{v}_{\mathsf{max}}]$. The range limits $\dot{v}_{\mathsf{min}} = -9\,\mathrm{m/s^2}$ and $\dot{v}_{\mathsf{max}} = 2\,\mathrm{m/s^2}$ reflect comfortable driving ($\dot{v}_t \geq \dot{v}_{\mathsf{max}}$) while allowing for the physical limit $\dot{v}_{\mathsf{min}}$ of the braking deceleration in safety-critical situations. The bounded action $a_t$ at time step $t$ in the range $[-1, 1]$ is mapped to the acceleration range by $\dot{v}_t = \min(j\dot{v}_{\mathsf{min}} a_t, \dot{v}_{\mathsf{max}})$.

The state space defines the observations that the vehicle can receive from the environment. To compute an optimal acceleration, the following vehicle is able to observe its own acceleration $\dot{v}_t$, its own speed $v_t$, the leader's speed $v_{t,l}$, and the (bumper-to-bumper) gap $g_t$ at time $t$. Note that these observations are identical to the input of the IDM (Treiber et al., 2000). Linear translation and scaling are used to reduce the dimensions and to bring all observations approximately into the same range. The Free-Driving Policy requires to observe just its own acceleration and speed, resulting in an observation vector at time step $t$ defined as

$$s_{t,\mathsf{free}} = \begin{pmatrix} \frac{v_t}{v_{\mathsf{des}}} \\ \frac{v_t - v_{\mathsf{min}}}{v_{\mathsf{max}} - v_{\mathsf{min}}} \end{pmatrix}, \tag{3}$$

where $v_{des}$ defines the desired speed. The Car-Following policy additionally needs to gain information about the dynamics of the leader vehicle, resulting in the observation vector

$$s_{t,follow} = \begin{pmatrix} \frac{v_t}{v_{\mathsf{des}}} \\ \frac{v_t - v_{\mathsf{min}}}{v_{\mathsf{max}} - v_{\mathsf{min}}} \\ \frac{v_{t,l} - v_t}{v_{\mathsf{des}}} \\ \frac{g_t}{g_{\mathsf{max}}} \end{pmatrix}, \tag{4}$$

where $g_{\mathsf{max}}$ is set to 200 m. Since at each time step, the Car-Following Policy needs to compute an acceleration value, even when there is no leader ahead or the leader is far away, $g_t$ is limited to $g_{\mathsf{max}}$ in simulation. This is a technical solution to avoid training for a bigger range of gaps.

### 2.5 Reward

The definition of the reward function is a crucial step in RL. In general, we have four objectives that have to be considered in car-following: First, the model needs to adhere to the desired speed $v_{\mathsf{des}}$. Second, comfortable driving in non-safety-critical situations has to be considered. The third objective is to be safe and accident-free. And finally, the follower needs to be keeping up with the leader vehicle, referred to as efficiency. In the following, we define different reward factors trying to reflect these objectives.

The first objective is to not exceed a desired speed $v_{\mathsf{des}}$, but also to accelerate if the desired speed $v_{\mathsf{des}}$ is not reached yet. On the contrary, if the current speed is higher than the desired speed, the agent needs to decelerate. This behavior can be modelled by the following reward factor

$$r_{t,\mathsf{speed}} = \begin{cases} \frac{v_t}{v_{\mathsf{des}}}, & \text{if } v_t \leq v_{\mathsf{des}}, \\ \frac{v_{\mathsf{des}}}{v_t}, & \text{otherwise.} \end{cases} \tag{5}$$

The second objective is comfortable driving. Therefore, we design a reward factor that motivates the agent to minimize the jerk, defined as

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

where $j_{\mathsf{comf}}$ 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, $r_{t,\mathsf{jerk}}$ values above unity tend to feel uncomfortable.

The most critical objective of the car-following policy is to reduce the crash risk and maintain a safe gap to the leader vehicle. Therefore, we design a reward factor that models the response in safety-critical situations by comparing the

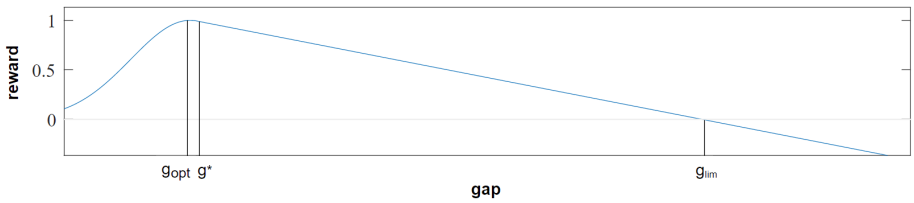


Figure 1: Factor $r_{t,\mathsf{gap}}$ of the reward function maximizing the reward for car-following with time gap $T$

kinematically needed deceleration (assuming an unchanged speed of the leader) with the comfortable deceleration $b_{\mathsf{comf}}$. This factor is defined as

$$r_{t,\mathsf{safe}} = \quad \tanh\left(\frac{b_{\mathsf{kin}} \quad b_{\mathsf{comf}}}{\dot{v}_{\mathsf{min}}}\right) \mathbb{1}(b_{\mathsf{kin}} > b_{\mathsf{comf}}) \tag{7}$$

with

$$b_{\mathsf{kin}} = \frac{(v_t \quad v_{t,l})^2}{g_t} \mathbb{1}(v_t > v_{t,l}) \tag{8}$$

the kinematic deceleration $b_{\mathsf{kin}}$ representing the minimum deceleration necessary to avoid a collision. The argument of the $\tanh(\,)$ function with the maximum possible deceleration ( $\quad 9\,\mathrm{m/s^2}$ 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 serves as a limitation for the reward value to the range $[\quad 1, 0]$. This has shown to make the learning process more stable based on own experiments. Notice that the case distinction in (7) ensures that this term is not activated in non-critical situations. The purpose of the factor $r_{t,\mathsf{safe}}$ is twofold: It motivates the follower vehicle to brake in safety-critical or near-crash situations. Furthermore, it motivates the follower vehicle also to break early in noncritical situations if the expected needed deceleration is above the comfortable level, in order to achieve a comfortable approaching of the leader vehicle.

Finally, we address the objective of efficiency by modelling a reward factor that motivates the agent to keep up with the leader vehicle, defined as

$$r_{t,\mathsf{gap}} = \begin{cases} \frac{\varphi f(g_t \quad g_{\mathsf{opt}})/g_{\mathsf{var}} g}{\varphi(0)}, & \text{if } g_t < g \ , \\ \frac{\varphi f(g_t \quad g_{\mathsf{opt}})/g_{\mathsf{var}} g}{\varphi(0)} \left(1 \quad \frac{g_t \quad g^*}{g_{\mathsf{lim}} \quad g^*}\right) & \text{otherwise,} \end{cases} \tag{9}$$

with $g_{\mathsf{opt}} = v_t T + g_{\mathsf{min}}$, $g_{\mathsf{var}} = 0.5 g_{\mathsf{opt}}$, $g_{\mathsf{lim}} = v_t T_{\mathsf{lim}} + 2 g_{\mathsf{min}}$, 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 $r_{t,\mathsf{gap}}$ is differentiable. Figure 1 illustrates the reward function for $r_{t,\mathsf{gap}}$, containing the parameter $g_{\mathsf{opt}}$, $g$ and $g_{\mathsf{lim}}$. 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 $g_{\mathsf{min}}$. Different values of $T$ result in different driving styles in a way that, for lower values of $T$ and $g_{\mathsf{min}}$, the driver follows more closely the leading vehicle resulting in a more aggressive driving style. 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 $T_{\mathsf{lim}}$ 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 chosen values for all reward parameters are listed in Table 2 in the appendix.

Finally, we model the final reward function for the Free-Driving and the Car-Following Policy at simulation time step $t$ as the weighted sum of the defined reward factors

$$r_{t,\mathsf{free}} = r_{t,\mathsf{speed}} + w_{\mathsf{jerk}} r_{t,\mathsf{jerk}}, \tag{10}$$

$$r_{t,\mathsf{follow}} = r_{t,\mathsf{safe}} + w_{\mathsf{gap}} r_{t,\mathsf{gap}} + w_{\mathsf{jerk}} r_{t,\mathsf{jerk}}, \tag{11}$$

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.

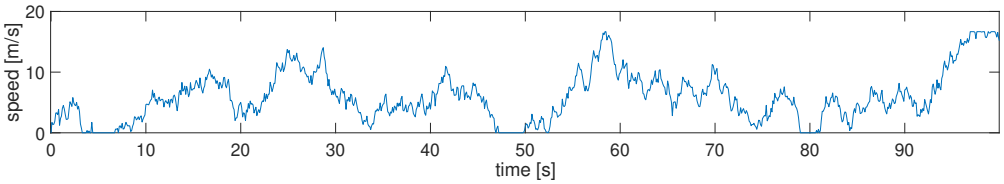


Figure 2: Example of a leading trajectory based on the parametrized Ornstein-Uhlenbeck process to train the RL agent

### 2.6 State update

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 Treiber and Kanagaraj (2015) as an efficient and robust scheme for integrating car-following models

$$v_{t+1} = v_t + \dot{v}_t \Delta t, \tag{12}$$

$$x_{t+1} = x_t + \frac{v_t + v_{t+1}}{2} \Delta t, \tag{13}$$

with $\Delta t$ corresponding to the simulation step size that is globally set to 100 ms.

## 3. Training environment

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 $[0, v_{\text{des}}]$. These specifications are sufficient to define the training environment for the Free-Driving Policy.

To train the Car-Following Policy, we additionally have to model the trajectory of the leader vehicle. As motivated in the introduction, we aim to train a generally applicable car-following model that is capable of handling all different kinds of car-following states. Therefore, we define the leading trajectory in training as a stochastic process to cover as much of the distribution of car-following states as possible. In particular, we use an Ornstein–Uhlenbeck process, defined as

$$\mathrm{d}v_{t,l} = \theta_2 \left(\mu_2 \quad v_{t,l}\right) \mathrm{d}t + \sigma_2 \mathrm{d}W_t, \tag{14}$$

with $\theta_2 = 0.132\,\mathrm{s}^{-1}$, $\mu_2 = 7.5\,\mathrm{m/s}$, $\sigma_2 = 3.847\,\mathrm{m/s}^{1.5}$ and $W_t$ denoting the Wiener process. The parameters have been designed to model the behavior of a real leader vehicle. For the numerical simulation, the Ornstein-Uhlenbeck process has been discretized using the Euler-Maruyama method (Okhrin et al., 2021). The resulting discretized process is defined as

$$v_l(n+1) = v_l(n) + \theta_2 \left(\mu_2 \quad v_l(n)\right) \Delta t + \sigma_2 \Delta W_n, \tag{15}$$

with

$$\Delta W_n \quad N(0, \Delta t). \tag{16}$$

Figure 2 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 $[0, 16.6\,\mathrm{m/s}]$. 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 2. The leader's trajectory also contains intervals where the speed is above the desired speed $v_{\text{des}} = 15\,\mathrm{m/s}$, e.g., around $t = 58\,\mathrm{s}$ and $t \in [97\,\mathrm{s}, 100\,\mathrm{s}]$, leading to free-driving situations. Furthermore, we parameterize the Ornstein-Uhlenbeck process such that the leading trajectory covers high acceleration and deceleration episodes at the limit of the physically possible. This high variability in acceleration produces an unpredictable leader behavior as well as a high coverage of different safety-critical events leading to good results regarding the generalization capabilities of the trained agent in unseen scenarios.

The initial space gap $g_0$ between both is set to 120 m simulating an initial episode of free traffic where the RL vehicle approaches its leader.

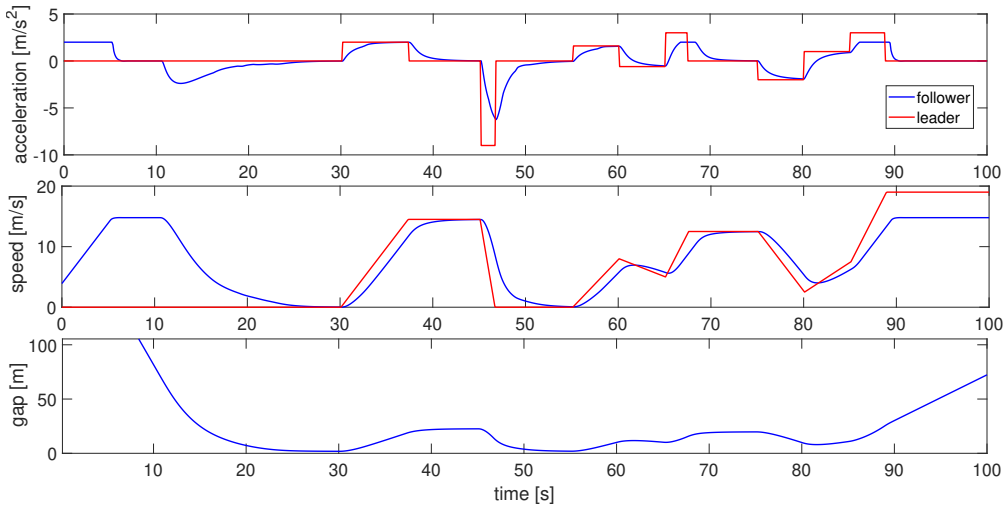


Figure 3: Response to an external leading vehicle speed profile.

## 4. Validation

We aim to validate our trained model regarding the defined objectives safety, efficiency, comfort, and adherence to desired speed based on different driving scenarios. We set the focus on scenarios that are not in the scope of the training data to prove the robustness of the model in different kinds of car-following events, especially in safety-critical situations. Moreover, we also validate the model against a criterion that was neither included in the training data nor in the reward functions: string stability.

### 4.1 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 3 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 $\dot{v}_{\mathsf{max}} = 2\,\mathrm{m/s^2}$ until the desired speed $v_{\mathsf{des}} = 15\,\mathrm{m/s}$ 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 $b_{\mathsf{comf}} = 2\,\mathrm{m/s^2}$ (transition between free driving and car-following) and comes to a standstill with a final gap of approximately $g_{\mathsf{min}} = 2\,\mathrm{m}$. 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 ($\dot{v}_l = \quad 9\,\mathrm{m/s^2}$) to a full stop ($t = 46\,\mathrm{s}$). At the time of the start of the emergency braking, the follower has nearly reached a steady following mode at the desired space gap $g_t = g_{\mathsf{min}} + v_t T$. 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 $\dot{v} \quad 5\,\mathrm{m/s^2}$. 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 $\quad 1.5\,\mathrm{m/s^3}$. 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 $g_{\mathsf{min}} + v_t T$ 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 $v_{\mathsf{des}} = 15\,\mathrm{m/s}$.

### 4.2 String stability

The second validation scenario, shown in Figure 5, consists of a leader based on the Ornstein-Uhlenbeck process from Section 3 that is followed by five vehicles, each controlled by the trained RL agent. The results show that traffic

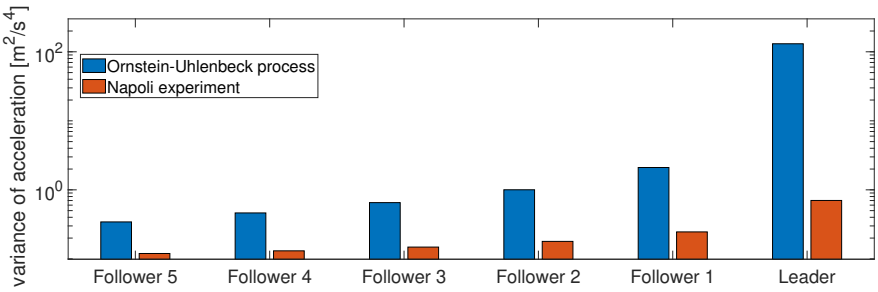


Figure 4: Comparison of the acceleration variance between leader and follower for a leader controlled by the Ornstein-Uhlenbeck process (blue bars) and the leading vehicle of the Napoli experiment (red).

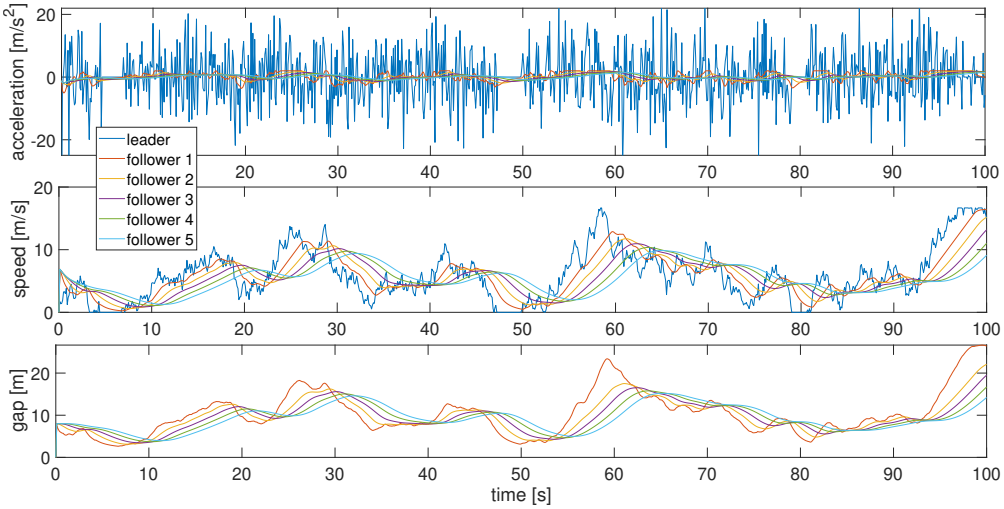


Figure 5: Response to a leader trajectory based on an Ornstein-Uhlenbeck process

oscillations can effectively be dampened with a sequence of trained followers, even if the leader shows large outliers in acceleration. Figure 4 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.

### 4.3 Response to a real leader trajectory

In a further scenario, the abilities of the RL strategy are evaluated with a real leader trajectory (Figure 6). This trajectory comes from the platoon driving experiments of Punzo et al. (2005) on urban and peripheral arterials in Napoli where high-precision distance data between the platoon vehicles were obtained. Similar to the experiment from Section 4.2, string stability and the reduction of the acceleration variance, shown by the red bars in Figure 4, 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.

Comparing the first three RL followers with the three followers of the real experiment, we notice that the RL followers drove more comfortably with less acceleration than the real drivers. This confirms the validity of our approach to not imitate real human driving behavior using deep learning methods but using RL to maximize an external reward function instead.

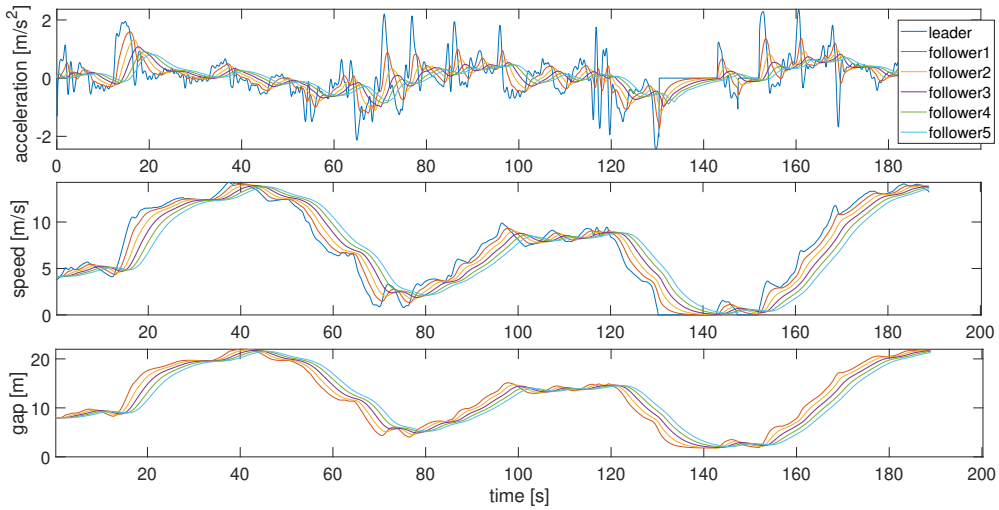


Figure 6: Response to a real leader trajectory

## 5. Discussion/Conclusion

This study presented a complete car-following model based on 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 a synchronized modular reinforcement learning architecture to decompose the multi-objective 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. We defined safety, efficiency, comfort, and adherence to desired speed as the objectives of car-following and used these objectives to define reward functions for each policy. Thereby, different driver characteristics can be achieved by adjusting the parameter of the reward function. To increase the model's robustness in car-following events that are not in the scope of the training, we propose a training environment based on Ornstein-Uhlenbeck processes. The usage of a stochastic process as the leader trajectory leads to high coverage of safety-critical events in training. Furthermore, the supply of learning data is unlimited.

For the validation of the trained agents, we simulated qualitatively different traffic scenarios based on both synthetic and real trajectory data that has not been seen in training. These scenarios include extreme 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. A reason for this favorable behavior is that, unlike classical car-following models, RL-based models even learn responses that are not explicitly contained in their specification. For example, the only reward component depending on the relative speed is the safety component (7) which only kicks in for large approaching rates. Still, the actual model also responds to small values of the relative speed since such behavior may prevent reaching a safety-critical situation within the prediction horizon.

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 what will be investigated in future work.

## References

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

Sooraj Bhat, Charles L Isbell, and Michael Mateas. On the difficulty of modular reinforcement learning for real-world partial programming. In *AAAI*, volume 1, pages 318–323, 2006.

Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics and Automation Letters*, 6 (4):7973–7980, 2021.

L. Chong, Montasir M. Abbas, and Alejandra Medina. Simulation of driver behavior with agent-based back-propagation neural network. *Transportation Research Record*, 2249:44 – 51, 2011.

Hongbo Gao, Guanya Shi, Kelong Wang, Guotao Xie, and Yuchao Liu. Research on decision-making of autonomous vehicle following based on reinforcement learning method. *Industrial Robot: the international journal of robotics research and application*, 2019.

Yaobang Gong, Mohamed Abdel-Aty, Jinghui Yuan, and Qing Cai. Multi-objective reinforcement learning approach for improving safety at intersections with adaptive traffic signal control. *Accident Analysis & Prevention*, 144: 105655, 2020. ISSN 0001-4575.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022.

David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

Liming Jiang, Yuanchang Xie, Xiao Wen, Danjue Chen, Tienan Li, and Nicholas G Evans. Dampen the stop-and-go traffic with connected and automated vehicles–a deep reinforcement learning approach. In *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–6. IEEE, 2021.

A. R. Kreidieh, C. Wu, and 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)*, pages 1475–1480, 2018.

Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

Yuan Lin, John McPhee, and Nasser L Azad. Comparison of deep reinforcement learning and model predictive control for adaptive cruise control. *IEEE Transactions on Intelligent Vehicles*, 6(2):221–231, 2020a.

Yuan Lin, John McPhee, and Nasser L Azad. Anti-jerk on-ramp merging using deep reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 7–14. IEEE, 2020b.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Ostap Okhrin, Michael Rockinger, and Manuel Schmid. Simulating the cir and heston processes: Matching the first four moments. *unpublished*, Jun 2021.

Vincenzo Punzo, Domenico Josto Formisano, and Vincenzo Torrieri. Nonstationary kalman filter for estimation of accurate and consistent car-following data. *Transportation research record*, 1934(1):2–12, 2005.

Xiaobo Qu, Yang Yu, Mofan Zhou, Chin-Teng Lin, and Xiangyu Wang. Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: a reinforcement learning based approach. *Applied Energy*, 257:114030, 2020.

Wouter J Schakel, Bart Van Arem, and Bart D Netten. Effects of cooperative adaptive cruise control on traffic flow stability. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 759–764. IEEE, 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

G. Tolebi, N. S. Dairbekov, D. Kurmankhojayev, and R. Mussabayev. Reinforcement learning intersection controller. In *2018 14th International Conference on Electronics Computer and Computation (ICECCO)*, pages 206–212, 2018.

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

Martin Treiber and Venkatesan Kanagaraj. Comparing numerical integration schemes for time-continuous car-following models. *Physica A: Statistical Mechanics and its Applications*, 419:183–195, Feb 2015. ISSN 0378-4371.

Martin Treiber and Arne Kesting. *Traffic Flow Dynamics*, chapter 10, page 158. Springer, 1 edition, 2013.

Martin Treiber and Arne Kesting. The intelligent driver model with stochasticity – new insights into traffic flow oscillations. *Transportation Research Part B: Methodological*, 117:613 – 623, 2018. ISSN 0191-2615.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930.

P. Wang, C. Chan, and A. de La Fortelle. A reinforcement learning based approach for automated lane change maneuvers. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1379–1384, 2018.

Pin Wang and Ching-Yao Chan. Autonomous ramp merge maneuver based on reinforcement learning with continuous action space. *arXiv preprint arXiv:1803.09203*, 2018.

Zhe Wang, Zhongyuan Tian, Jiang Xu, Rafael K. V. Maeda, Haoran Li, Peng Yang, Zhehui Wang, Luan H. K. Duong, Zhifei Wang, and Xuanqi 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)*, pages 684–689, 2017.

Zhi Xu, Shuncheng Liu, Ziniu Wu, Xu Chen, Kai Zeng, Kai Zheng, and Han Su. Patrol: A velocity control framework for autonomous vehicle via spatial-temporal reinforcement learning. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia*, 2021.

Yi-Tung Yen, Jyun-Jhe Chou, Chi-Sheng Shih, Chih-Wei Chen, and Pei-Kuei Tsung. Proactive car-following using deep-reinforcement learning. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

Wu Yuankai, Huachun Tan, Jiankun Peng, and Bin Ran. A deep reinforcement learning based car following model for electric vehicle. *Smart City Application*, 2, 09 2019.

Yi Zhang, Ping Sun, Yuhan Yin, Lin Lin, and Xuesong Wang. Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1251–1256, 2018.

Mofan Zhou, Xiaobo Qu, and Xiaopeng Li. A recurrent neural network based microscopic car following model to predict traffic oscillation. *Transportation Research Part C: Emerging Technologies*, 84:245–264, 2017. ISSN 0968-090X.

Meixin Zhu, Xuesong Wang, and Yinhai Wang. Human-like autonomous car-following model with deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 97:348–368, 2018. ISSN 0968-090X.

Meixin Zhu, Yinhai Wang, Ziyuan Pu, Jingyun Hu, Xuesong Wang, and Ruimin Ke. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies*, 117:102662, 2020. ISSN 0968-090X.

## Appendix A. Modular RL architecture and model parameters

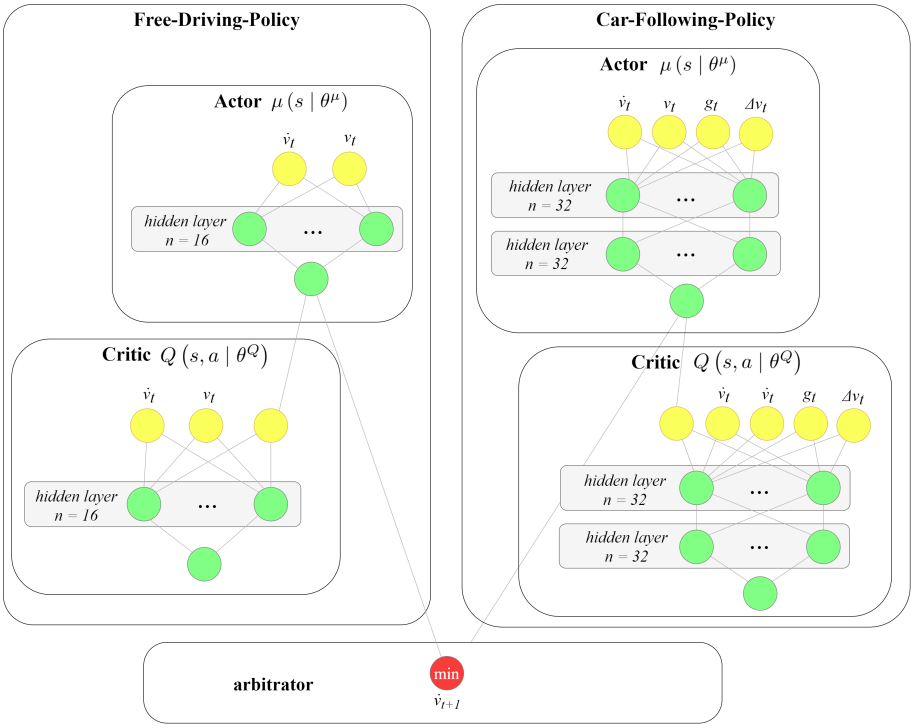


Figure 7: Synchronized modular RL architecture with actor and critic networks of both policies.

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 $\theta_1$ | 0.15 | 0.15 |
| Ornstein-Uhlenbeck $\sigma_1$ | 0.2 | 0.2 |
| Number of hidden layers | 1 | 2 |
| Neurons per hidden layer | 16 | 32 |
| Soft target update $\tau$ | 0.001 | 0.001 |

Table 2: RL agent parameters and default values

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