actuators

MDPI

Article

Learning-Based Cooperative Adaptive Cruise Control

Jonas Mirwald [1], Johannes Ultsch [1], Ricardo de Castro [2] and Jonathan Brembeck [1,*]

[1] Institute of System Dynamics and Control, Robotics and Mechatronics Center, German Aerospace Center (DLR), Münchener Str. 20, 82234 Wessling, Germany; jonas.mirwald@dlr.de (J.M.); johannes.ultsch@dlr.de (J.U.)
[2] Department of Mechanical Engineering, University of California, Merced, CA 95343, USA; rpintodecastro@ucmerced.edu
* Correspondence: jonathan.brembeck@dlr.de; Tel.: +49-8153-28-2472

Abstract: Traffic congestion and the occurrence of traffic accidents are problems that can be mitigated by applying cooperative adaptive cruise control (CACC). In this work, we used deep reinforcement learning for CACC and assessed its potential to outperform model-based methods. The trade-off between distance-error minimization and energy consumption minimization whilst still ensuring operational safety was investigated. Alongside a string stability condition, robustness against burst errors in communication also was incorporated, and the effect of preview information was assessed. The controllers were trained using the proximal policy optimization algorithm. A validation by comparison with a model-based controller was performed. The performance of the trained controllers was verified with respect to the mean energy consumption and the root mean squared distance error. In our evaluation scenarios, the learning-based controllers reduced energy consumption in comparison to the model-based controller by 17.9% on average.

Keywords: cooperative adaptive cruise control; C2C communication; deep reinforcement learning; Modelica vehicle modeling

Citation: Mirwald, J.; Ultsch, J.; de Castro, R.; Brembeck, J. Learning-Based Cooperative Adaptive Cruise Control. *Actuators* **2021**, *10*, 286. https://doi.org/10.3390/act10110286

Academic Editor: Hai Wang

Received: 17 September 2021
Accepted: 21 October 2021
Published: 26 October 2021

1. Introduction

Cooperative adaptive cruise control (CACC) contributes to an efficient, higher-density traffic flow, especially on highways. Furthermore, since human errors are a major reason for traffic accidents, removing the driver from the driving task by using CACC in specific cases improves traffic safety [1]. In addition, the aerodynamic drag is reduced when driving in a vehicle platoon, therefore CACC-enabled platooning is an effective way to reduce fuel consumption [2].

CACC is an extension of adaptive cruise control (ACC). The basic cruise control (CC) enables the driver to define a certain velocity set point, which is then automatically tracked by the vehicle. ACC extends this approach by detecting obstacles in front of the vehicle and adjusting the velocity accordingly. CACC is a further extension using car-to-car (C2C) communication. With the exchange of information between the vehicles, the motion of the other vehicles can be anticipated. This allows a reduction in distance to the preceding vehicle, while maintaining driving comfort and safety. For full autonomous platooning, CACC must be extended by a lateral control that enables lane changing and lane keeping [3].

To allow safe operation, string stability is one of the most important control requirements for CACC [1]. If the platoon is string stable, fast deceleration of a preceding vehicle and a diminished distance to the following vehicle do not amplify downstream [4]. If this is not the case, the amplification may cause a following vehicle further downstream to stop entirely or crash into the preceding vehicle.

In the past years, there have been great advances in the development of artificial intelligence, especially regarding deep reinforcement learning (deep RL, DRL) [5]. In RL,

an agent is trained by interaction with an environment such that a given goal is achieved. In DRL, this agent is approximated with a neural network. Many of these recent advances rely on model-free approaches, in which the agent does not have access to the environment's mathematical model.

The application of DRL to highly complex games (Atari [6], Go [7], Dota 2 [8]) has especially sped up the research, but DRL is also applied to control problems ranging from lateral and longitudinal path following control [9] to power grid control [10], to active flow control [11], and to traffic signal control [12].

Classical control methods can already tackle multiple aspects of CACC. Model predictive control (MPC) was used to create energy-efficient acceleration profiles for platoon vehicles that also consider the string stability of the platoon [13]. Communication losses were considered for CACC control by using, e.g., a Kalman filter to estimate the preceding vehicle's acceleration during short intervehicle communication outages [14].

However, the incorporation of a string stability condition may lead to higher complexity when tuning the MPC controller [15]. In addition, larger prediction horizons lead to an increased computational effort, which lowers the usability of MPC controllers for real-world applications [16].

Compared to classical control methods, the model-free DRL's robustness may be difficult to guarantee [17], but there are also multiple advantages, including:

1. There is no need to create a (simplified) mathematical system model, as the DRL agent can directly learn from a real system or a simulated high-fidelity model [18];
2. All relevant information can be directly fed into the agent's state vector, and the system's constraints are learned by the agent's neural network [6,19], both without necessarily creating preprocessing structures, and;
3. The deployment of the trained agent's neural network is computationally efficient [16].

For CACC, this means that DRL has the potential to outperform classical methods regarding the sim-to-real-gap, the application complexity for the control engineer, as well as the required performance of the deployment hardware.

DRL has already been assessed for some aspects of CACC. In [16], the authors investigated DRL for CC and compared it to MPC regarding predictive velocity tracking. It was shown to achieve similar performance with a computational effort up to 70-fold lower. The authors of [20] compared DRL to MPC for ACC and also found it to be comparable in performance. DRL provided lower costs (combined penalized error, control signal amplitude, and vehicle jerk) than MPC when facing model uncertainties (in the context of sim-to-real-gap) due to its generalization capabilities.

Reference [21] applied DRL on the ACC as well as the CACC problem, and studied the effect of discrete control outputs on a set point distance. However, the discrete control outputs led to intense oscillations in the velocity and acceleration trajectory. The authors of [22] augmented the approach by using continuous actions. To cope with possible issues regarding safety and robustness, they also investigated the possibility to replace a DRL agent that directly controlled the vehicle with a model-based DRL agent that only controlled the headway parameter of a model-based controller. They concluded that the model-based DRL agent was safer and more robust.

In [23], a supervisor network was used to augment a RL agent to increase its performance. The combined output of the supervisor and the agent was applied to the CACC system, which led to a reduced distance error compared to a linear controller.

In addition, a multiagent RL (MARL) can be used to consider the dynamics of a whole vehicle platoon already during training [24].

The main contribution of this work is the application of DRL to CACC with continuous action and state space, while also considering

1. Energy minimization;
2. A string stability constraint;
3. Preview information in the communication, and;
4. The effect of burst errors on the communication.

Previous research examined some of these aspects (e.g., energy minimization was considered in [19], preview information in [16], and communication errors in [24]), but not the combination of all four. To the authors' best knowledge, this is one of the first works to apply a string stability constraint in the presented way to DRL-based CACC. Additionally, a high model-complexity was incorporated: a high-fidelity model was used to model energy losses in the powertrain, and the platoon leader vehicle's motion was modeled according to a real-world dataset. This work was based on one of the author's unpublished master's thesis [25].

The remainder of this work is organized as follows: in Section 2, the CACC problem and the platoon model are introduced. Section 3 explains the basics of DRL and the application to the CACC problem. In Section 4, the training of the DRL agents is described, and the comparison to a model-based controller is carried out. Section 5 summarizes the findings and gives an outlook.

2. Problem Formulation

The aim of CACC control is to ensure that the vehicles that form a platoon:

1. Stay at a safe distance from each other to prevent collisions, but;
2. Drive close enough to the preceding vehicle such that the air drag is decreased (thereby decreasing energy consumption) and highway capacity is increased;
3. Consider passenger comfort by reducing vehicle jerk, and;
4. Drive in a string-stable way.

Figure 1 shows the follower vehicle with index i driving behind the preceding vehicle with index $i-1$, separated by the bumper-to-bumper distance d_i. Both feature their corresponding position x_i, velocity v_i, and acceleration a_i. The entire platoon is composed of m vehicles.

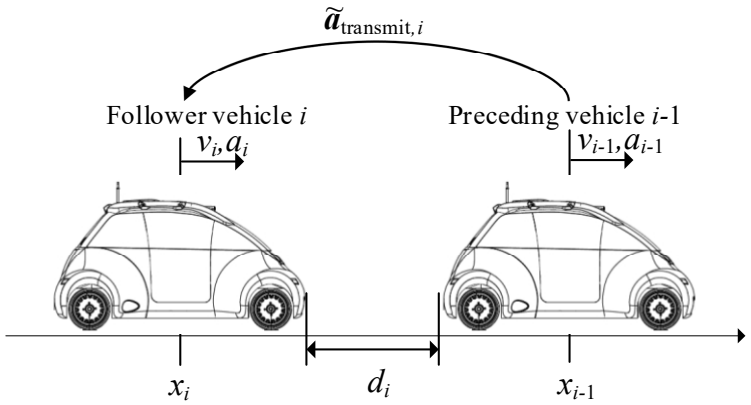

Figure 1. Platooning linear model with information flow [26].

The target distance d_i^* to the preceding vehicle may be decribed via a constant spacing or constant time-headway [1], with constant time-headway being defined as [26]:

$$d_i^* = r_{c,i} + t_{h,i} v_i. \tag{1}$$

where $r_{c,i}$ denotes the distance at standstill and $t_{h,i} v_i$ denotes the variable distance, adjusted by the time-headway $t_{h,i}$. The deviation from the target distance is described by the distance error:

$$e_i = d_i - d_i^*. \tag{2}$$

Analyzing string stability is more difficult in DRL than in a model-based approach, because the DRL problem is not typically designed in a frequency domain, where string

stability usually is easier to analyze [26,27]. In [28], it was shown that string stability can be guaranteed as follows:

$$\sup_{\omega} \left| \frac{a_i\ (j\omega)}{a_{i-1}\ (j\omega)} \right| \leq 1,\ 2 \leq i \leq m, \tag{3}$$

where $a(j\omega)$ is the acceleration of the respective vehicle in frequency domain.

2.1. Communication Model

Regarding the platooning C2C information flow, a predecessor-following topology (cf. [3]) was adopted: in such a topology, information is sent one way from the preceding vehicle to the following vehicle (cf. Figure 1).

For modeling the communication, an appropriate distance for transmitting data and a sufficient signal bandwidth was assumed. For a specific prediction horizon N_k, the preceding vehicle $i-1$ transmits its current acceleration and its predicted accelerations over a N_k-window $\boldsymbol{a}^{\mathrm{T}}_{\mathrm{transmit},i-1,k} = \left[a_{i-1,k},\ a_{i-1,k+1},\ \ldots, a_{i-1,k+(N_k-1)}\right]$. For $N_k = 1$, only the current acceleration is transmitted. The follower vehicle receives the delayed acceleration $\widetilde{\boldsymbol{a}}^{\mathrm{T}}_{\mathrm{transmit},i,k} = \left[\widetilde{a}_{i,k},\ \widetilde{a}_{i,k+1},\ \ldots, \widetilde{a}_{i,k+(N_k-1)}\right]$ with an assumed delay of Δ_i for vehicle i (cf. Table 1).

Table 1. Vehicle model parameters and general CACC problem parameters.

Parameter	Value	Parameter	Value
Δt	0.1 s	M_{nom}	4×160 Nm
Δ_i	0.1 s	T_M	0.1 s
a_{invalid}	-10 m/s^2	k_M	8000
U_{batt}	322.4 V	A_x	1.232 m^2
R_{batt}	0.54 Ω	c_{w}	0.3
$n_{\mathrm{cells,s}}$	90	$c_{d,1}$	17.58
U_{cell}	3.582 V	$c_{d,2}$	34.03
R_{cell}	0.006 Ω	ρ_{air}	1.25 Ns^2/m^4

In this communication, burst errors due to noise may occur, and a sequence of successive transmissions $\widetilde{\boldsymbol{a}}_{\mathrm{transmit},i,k}$ is not received. The burst errors are modeled with a Markov chain according to the Gilbert–Elliot model [29] (cf. [30], Figure 2). The Markov chain consists of a state r, which can receive transmissions, and a state l, in which transmissions are unavailable. According to the Markov property, the transition to the next state only depends on the present state. This means that for state r, it remains in state r with probability $p = p_r$ and $0 < p_r < 1$, or leaves it with $p = 1 - p_r$. The same holds true for state l with $p = p_l$ and $0 < p_l < p_r$. The Markov chain was implemented as an automaton, starting in the receiving state. The probabilities used for the automaton in case of the occurrence of burst errors were taken from [30]. They parametrized a low communication quality, and can be found in Table 2. A parametrization for perfect communication quality is also given.

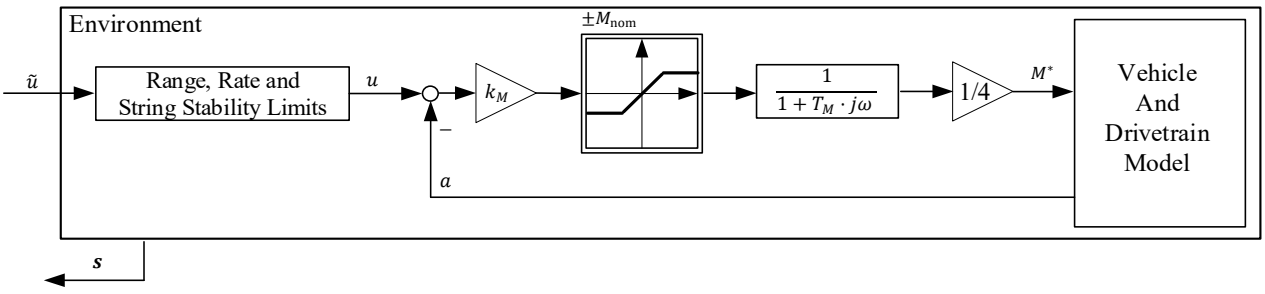

Figure 2. Block diagram of the environment.

Table 2. Communication parametrization regarding the communication quality and the probability of burst errors.

Communication Quality	p_r	p_l
Perfect	1	0
Low	0.8	0.75

Reference [30] suggested the use of a buffer when a sequence of prediction values is transmitted. The buffer is referred to as $\widetilde{a}_{\text{buffer},i,k}$. For each time step k, the received transmitted acceleration $\widetilde{a}_{\text{transmit},i,k}$ is saved in the buffer: $\widetilde{a}_{\text{buffer},i,k} = \widetilde{a}_{\text{transmit},i,k}$. If a burst error occurs, no $\widetilde{a}_{\text{transmit},i,k}$ is received. In this case, the first element of the buffer no longer contains valid information and the resulting gap in the buffer is filled with a placeholder a_{invalid} (cf. Table 1). a_{invalid} is an invalid acceleration value that was chosen to be large enough not to be attainable in our scenario, such that it could not occur in regular communication, and was interpretable as a communication error by the DRL agent.

2.2. Vehicle Model

The single vehicle was created as a two-track model in Modelica [31] using Dymola [32] and the planar mechanics library [33]. Modelica is an object-oriented open source modeling language for multiphysical (e.g., mechanical and electrical) systems. The vehicle model represented the properties of the ROboMObil (ROMO) [34], which is the robotic full x-by-wire research vehicle of the German Aerospace Center (cf. Figure 1 for two ROMOs forming a platoon). The center of gravity (used to calculate the vehicle's position x_i; cf. Figure 1) was assumed to be at the geometric center of the rectangle spanned by the track width and the wheelbase.

As wheel models, four dry-friction slip-based wheels of the planar mechanics library [33] were used and connected to fixed translation elements. The slip-based wheels limited the amount of propulsion force dependent on the normal force and the friction coefficient.

The energy consumption of the single vehicle mainly depended on the variable air drag and the electric powertrain.

ROMO's surface front A_x, air drag coefficient c_w, and the air density ρ_air were chosen according to [35] (cf. Table 1). Reference [36] stated that the air drag coefficient is dependent on the distance $d = d_i$ to the preceding vehicle: When driving close to the preceding vehicle, the air drag coefficient is reduced. This reduced air drag leads to less power being necessary for driving. To model the variable air drag, a nonlinear approximation was used, as suggested by [37]:

$$c_d(d) = c_\text{w} \cdot \left(1 - \frac{c_{d,1}}{c_{d,2} + d}\right), \tag{4}$$

where $c_{d,1}$ and $c_{d,2}$ are variable air drag function coefficients. The parameters were obtained by performing a linear regression on the experimental data given in [38]. The variable air drag force can then be calculated as:

$$F_\text{aer}(d, v) = \frac{1}{2} \cdot c_d(d) \cdot \rho_\text{air} \cdot A_x \cdot v^2. \tag{5}$$

The battery of one single vehicle [39] consist of $n_{\text{cells,s}}$ cells connected in series (cf. Table 1). The total pack voltage U_batt and total pack resistance R_batt were assumed to be constant during experiments (at 100% state-of-charge). To account for battery losses, we considered Ohmic heat losses in each cell (each cell with respective voltage U_cell and resistance R_cell).

Each of the four wheels was powered by an in-wheel motor, which was modeled as an open-loop controlled permanent magnet synchronous machine [35,39,40]. This allowed us to calculate the vehicle's power consumption $P = P_i$ as:

$$P = P_{\text{mech}} + P_{\text{losses}}. \tag{6}$$

where P_{mech} is the summed mechanical power of all electric motors. For each j-th motor ($j \in \{1,2,3,4\}$), $P_{\text{mech},j} = M_j \cdot \omega_{\text{m},j}$, with motor torque M_j and mechanical rotor angular velocity $\omega_{\text{m},j}$. P also considered the powertrain losses P_{losses}, which consisted of the Ohmic heat losses in the battery and the losses in the electric motors (i.e., inverter losses, copper losses, iron losses, and mechanical friction losses).

A proportional controller was used as the low-level acceleration control (with a high gain k_M), which controlled the torque set point M^* for the electric motors. The torque was limited to $\pm M_{\text{nom}}$ (nominal torque, cf. Table 1), so that the physical constraints of the system were not violated. Additionally, a first-order low pass filter was applied to the acceleration control's output with a time constant T_M to achieve an actuator lag.

The block diagram in Figure 2 gives an overview of the used vehicle model with respect to the RL framework discussed in Section 3 (cf. Section 3.2 for more information on action u, denormalized neural network output $\widetilde{u}$, and the respective range, rate and string stability limits; cf. Section 3.3 for more information on feedback state $\boldsymbol{s}$).

2.3. Leader Vehicle Motion Model

For modeling the platoon's leader vehicle motion, trajectory data from the Next Generation Simulation (NGSIM) dataset was used [41]. In this dataset, positions of vehicles were measured with multiple synchronized cameras at different locations in the USA (Emeryville's Interstate 80; Los Angeles' Route 101; and Lankershim Boulevard, Atlanta's Peachtree Street) at two to three specific times of the day. In doing so, it was aimed at portraying different amounts of traffic and thereby different vehicle trajectories during the buildup of congestion, during transition between congested and uncongested traffic, as well as during full congestion.

The NGSIM data must be postprocessed due to measurement errors [42] to create consistency between velocity and acceleration signals. In this work, this was done in a multistep approach by:

1. Removing acceleration outliers ($a > \left|30\ \text{m/s}^2\right|$) by applying a natural cubic spline interpolation on the velocity profile;
2. Reducing the velocity profile noise by applying a first-order low-pass Butterworth filter (cutoff frequency of 0.5 Hz); and
3. Removing implausible accelerations ($a > 5\ \text{m/s}^2$ or $a < -8\ \text{m/s}^2$) also by applying a natural cubic spline interpolation on the velocity profile.

Finally, if implausible accelerations were still present or if velocities higher than ROMO's nominal velocity limit $v_{\text{nom}} = 27.4$ m/s occurred, the trajectory was removed from the set of valid trajectories. After doing so, the resulting set consisted of 312 trajectories. It was permutated and then split into training (70%) and validation (30%) sets.

3. Reinforcement-Learning-Based Cooperative Adaptive Cruise Control

In this work, we used reinforcement learning to solve the control problem. A Markov decision process (MDP) described the probabilistic basis of RL, and was used to derive the algorithms to find the optimal action in each time step. An MDP consists of an agent and the environment, as shown in Figure 3. In each discrete time step k, the agent perceives the current state $\boldsymbol{s}_k \in \mathcal{S}$ ($\mathcal{S}$ being the set of all valid states) of the environment and performs the corresponding action $\boldsymbol{u}_k \in \mathcal{A}$ ($\mathcal{A}$ being the set of all valid actions) [5]. If a reward $R_{k+1} \in \mathcal{R} \subset \mathbb{R}$ is assigned according to the action $\boldsymbol{u}_k$ and the resulting state $\boldsymbol{s}_{k+1}$, the MDP can be called a Markov reward process (MRP) [43]. The MRP is defined to begin with state $\boldsymbol{s}_0$ and action $\boldsymbol{u}_0$, but without an initial reward. The first reward R_1 is assigned after the

transition to the state s_1. From a control theory's perspective, the agent can be seen as controller, the environment as plant, and the action as control signal.

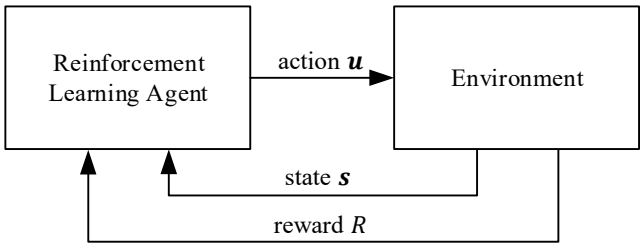

Figure 3. Agent-environment interaction in a Markov decision process [5].

A sequence beginning with s_0, u_0, R_1, s_1, u_1, R_2, s_2, u_2, ... and ending with a terminal state s_T is called an episode τ [5]. After reaching s_T, no further action can be taken and a reset to a starting state occurs. In this work, a fixed interval between action and new state $\Delta t = \Delta_i$ (cf. Table 1) was assumed, yielding the time $t = k \cdot \Delta t$ with $k \in \mathbb{N}^+$.

For the MDP, the Markov property also held true (cf. Section 2.1).

3.1. Policy Optimization

The policy π describes which action an agent takes given a certain state. This can be described with a stochastic policy $\pi : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ as in [43]:

$$u_k \sim \pi(\cdot \mid s_k). \tag{7}$$

The agent tries to maximize the discounted return G_k, which is the sum of rewards that are received until the terminal state is reached, each multiplied by a discount factor $\gamma \in [0, 1]$ [5]:

$$G_k R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \cdots + \gamma^{T-k-1} R_T. \tag{8}$$

In the case of $\gamma = 1$, the discounted return was just referred to as return in this work. The state-value function is defined as the expected discounted return for a state s when a given policy π is applied:

$$V_\pi(s) \mathbb{E}_\pi[\, G_k \mid s_k = s \,], \text{ for all } s \in \mathcal{S}. \tag{9}$$

The goal of RL is to find the optimal policy π^* that optimizes the expected discounted return for a specific episode τ (with the actions of τ being taken according to π):

$$\pi^* = \arg\max_\pi \mathop{\mathbb{E}}_{\tau \sim \pi}[G(\tau)]. \tag{10}$$

Regarding DRL, this work used the proximal policy optimization (PPO) algorithm with clipping [44,45] to calculate neural-network-based approximations of π^* and V_π. The weights of the neural network [46] π are referred to as $\boldsymbol{\theta}$, $s_k \in \mathcal{S} \subset \mathbb{R}^n$ and $u_k \in \mathcal{A} \subset \mathbb{R}^m$.

PPO uses gradient ascent to optimize $\boldsymbol{\theta}$, such that the obtained discounted returns following a state-action tuple (s, u) are more probable. These obtained discounted returns are described with the estimated advantage $\hat{A}_{k,\boldsymbol{\theta}} = G_k - V_\pi(s_k)$ [5,44,47]. PPO does not directly optimize the policy, but it uses an objective function $L\big(s, u, \boldsymbol{\theta}_i, \boldsymbol{\theta}, \hat{A}_{\boldsymbol{\theta}_i}\big)$ [44]:

$$\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{s,u \sim \pi}\big[\, L\big(s, u, \boldsymbol{\theta}_i, \boldsymbol{\theta}, \hat{A}_{\boldsymbol{\theta}_i}\big)\big], \tag{11}$$

with:

$$L\big(s, u, \boldsymbol{\theta}_i, \boldsymbol{\theta}, \hat{A}_{\boldsymbol{\theta}_i}\big) = \min\big(r_o(\boldsymbol{\theta})\hat{A}_{\boldsymbol{\theta}_i},\ g(r_o(\boldsymbol{\theta}), \epsilon)\hat{A}_{\boldsymbol{\theta}_i}\big), \tag{12}$$

where $r_o(\boldsymbol{\theta})$ is the probability ratio $r_o(\boldsymbol{\theta}) = \frac{\pi(u|s,\boldsymbol{\theta})}{\pi(u|s,\boldsymbol{\theta}_i)}$ and g is the clipping function. The clipping function limits the changes of $r_o(\boldsymbol{\theta})$ to $[1 - \epsilon, 1 + \epsilon]$. Thereby, the objective func-

tion L creates an incentive for only small changes of $\boldsymbol{\theta}$, which increases training stability compared to a basic DRL algorithm (in this work, training stability meant a monotonic increase of the obtained average returns during training).

The basic principle of operation of PPO is shown in Algorithm 1. As for the policy, the update of the value function V_π (step 11) can be performed via gradient descent, such that the objective function L is optimized. Policy and value function updates are repeated in multiple passes until the trained policy's behavior converges, which means until the obtained returns can approximately no longer be increased.

Algorithm 1. Basic PPO principle of operation [44,45,47].

1: Input: initial policy parameters $\boldsymbol{\theta}$, initial state-value parameters
2: **repeat:**
3: **for** each process $\mathcal{P}_i$ with $i = 1, \ldots, n_\mathcal{P}$:
4: Generate $T_\mathcal{P}$ transitions $\{\boldsymbol{s}_k, \boldsymbol{u}_k, R_{k+1}\}$ with $\pi(\cdot|\cdot)$
5: **for** each transition k with $k = 1, \ldots, T_\mathcal{P}$:
6: Compute G_k
7: $\hat{A}_{k,\boldsymbol{\theta}} = G_k - V_\pi(\boldsymbol{s}_k)$ [1]
8: **end for**
9: **end for**
10: Update the policy by maximizing the objective function L in n_e training epochs, sampling a minibatch (with $n_\mathcal{P} \cdot T_\mathcal{P} / n_\mathrm{mini}$ transitions) in each epoch
11: Update the value function
12: **until** convergence
13: Output: trained policy π, trained state-value function V_π

[1] Different advantage term calculations are possible.

The computational time of the training is decreased by using multiprocessing with $n_\mathcal{P}$ processes. For this, each process $\mathcal{P}_i$ is used to generate $T_\mathcal{P}$ transitions $\{\boldsymbol{s}_k, \boldsymbol{u}_k, R_{k+1}\}$ with a shared policy $\pi(\cdot|\cdot)$.

The policy is updated in n_e passes (called training epochs) for each pass of the PPO algorithm. For each update of $\boldsymbol{\theta}$ in an epoch, a minibatch consisting of multiple transitions is sampled. The minibatch is a subset of the batch of $n_\mathcal{P} \cdot T_\mathcal{P}$ transitions that is available in each pass of the PPO algorithm. The size of the minibatch is defined by dividing the size of the batch by the minibatch scaling factor n_mini. The number of passes until the algorithm converges is estimated by defining a fixed training length, which means the total number of generated transitions after which the training is ended.

3.2. Actions

The agent controls a follower vehicle (i.e., it acts with the environment) by setting a target acceleration $u_{i,k}$. To improve passenger comfort, a jerk limit is introduced to enforce action smoothness:

$$\left|\Delta u_{i,k}\right| \leq \Delta u_\mathrm{lim}, \tag{13}$$

with $\Delta u_\mathrm{lim} = 0.5\ \mathrm{m/s^2}$ and $\Delta u_{i,k} = \widetilde{u}_{i,k} - u_{i,k-1}$. The action $\widetilde{u}_{i,k}$ is the denormalized output of the policy π_i. Furthermore, only accelerations that are within the physical limits of the car are allowed to be set:

$$-8\ \mathrm{m/s^2} \leq u_{i,k} \leq 5\ \mathrm{m/s^2}. \tag{14}$$

In this work, the following two assumptions were made:

1. $a_{i,k} \approx u_{i,k}$,
2. $a_{i-1,k} \approx \widetilde{a}_{i,k}$.

To satisfy assumption 1, the low-level acceleration controller was assumed to provide an accurate tracking performance. To satisfy assumption 2, the acceleration of the preceding vehicle $a_{i-1,k}$ was assumed to be approximately equal to the delayed acceleration $\widetilde{a}_{i,k}$ that

was received by the follower vehicle (cf. Section 2.1). When we transferred the string stability concept of (3) to time domain with:

$$\left|a_{i,k}\right| < \gamma \cdot \max_{k_{\mathrm{ss}} \in [k-N_{\mathrm{ss}},\, k]} \left\{ \left|a_{i-1,k_{\mathrm{ss}}}\right| \right\}, \tag{15}$$

this allowed us to represent the string stability condition as:

$$\begin{aligned} \left|u_{i,k}\right| &< \gamma \cdot \max \left\{ u_{\mathrm{lim,min}},\, \max_{k_{\mathrm{ss}} \in [k-N_{\mathrm{ss}},\, k]} \left\{ \left|a_{i-1,k_{\mathrm{ss}}}\right| \right\} \right\} \\ &< \gamma \cdot u_{\mathrm{ss},i,k}. \end{aligned} \tag{16}$$

In (15), the acceleration of the follower vehicle a_i must be smaller than $\gamma \in (0,\, 1)$ times the maximum acceleration of the preceding vehicle within a string stability time window N_{ss} for every time step k. Thus, the follower vehicle is assumed to mimic the driving behavior of the preceding vehicle. If it does so and the accelerations are mostly attenuated downstream, string stability is guaranteed to a given extent. The transfer of the string stability concept (3) to time domain (15) was proposed in a similar way in [28].

Equation (16) makes the implementation of a string-stability condition easier, as it is not necessary to abort the episode due to a state constraint violation: only an additional control limit must be added. The resulting maximum from (16) is referred to as string-stability limit $u_{\mathrm{ss},i,k}$. Furthermore, only valid values of $\widetilde{\boldsymbol{a}}_{\mathrm{transmit},i,k}$ are taken into consideration, and $u_{\mathrm{ss},i,k}$ cannot be below $u_{\mathrm{lim,min}} = 0.1\ \mathrm{m/s^2}$. $u_{\mathrm{lim,min}}$ gives the agent the possibility to counteract slight perturbations.

In this work, $N_{\mathrm{ss}} = 20$ and $\gamma = 0.999$ were chosen. With the fixed time interval Δt (cf. Table 1) between two steps k and $k-1$, this meant that the current acceleration and the received accelerations from the past two seconds were taken into consideration.

3.3. Feedback State

The design of the CACC DRL environment considers a two-vehicle platoon. The motion of the first vehicle—the leader—followed a reference trajectory obtained from the NGSIM dataset (cf. Section 2.3). The motion of the second vehicle—the follower—was controlled by the DRL agent. The follower vehicle was referred to with index $i = 2$, the leader vehicle with index $i = 1$. The agent state vector $\boldsymbol{s}_{2,k}$ of the follower vehicle for time step k was chosen as

$$\boldsymbol{s}_{2,k} = [v_{2,k},\, a_{2,k},\, d_{2,k},\, \Delta v_{2,k},\, e_{2,k},\, P_{2,k}, u_{2,k-1},\, u_{2,k-2},\, \widetilde{\boldsymbol{a}}_{\mathrm{buffer},2,k},\, u_{\mathrm{ss},2,k}]^{\mathrm{T}}. \tag{17}$$

The states were partially chosen with respect to (1), with $v_{2,k}$ being the velocity of the vehicle and $a_{2,k}$ being the acceleration of the vehicle. $d_{2,k}$ is the distance to the preceding vehicle:

$$d_{2,k} = x_{1,k} - x_{2,k} - \frac{l_1}{2} - \frac{l_2}{2}, \tag{18}$$

with $x_{i,k}$ being the position of the respective vehicle and l_i its respective vehicle length. $\Delta v_{2,k} = v_{1,k} - v_{2,k}$ is the discrete approximation of the distance dynamics $\dot{d}_2$ at time $t = k \cdot \Delta t$. $e_{2,k}$ is the distance error (cf. (2)) and $P_{2,k}$ the power consumption at time step k (cf. (6)). Even though we did not investigate if this state selection fulfilled the Markov property, assembling the feedback state this way provided the agent with rich information about the environment.

The state $\boldsymbol{s}_{i,k}$ was augmented with the history of past control values (as suggested by [48]) for learning a smooth control action. For this purpose, the actions from the prior two time steps $u_{2,k-1}$ and $u_{2,k-2}$ were provided.

The state $\boldsymbol{s}_{2,k}$ was normalized before it was fed into the policy π_2 to ensure the agent considered each state component equally.

3.4. Reward Function

The control goal of string stability is targeted by the action limit (16), while the jerk limit (13) ensures at least a certain degree of passenger comfort. The control objective of maintaining a safe distance while reducing air drag is incorporated into the CACC environment through the reward function.

To enhance the passenger comfort besides the already constrained change in control value $\Delta u_{i,k}$ (cf. (13)), $\Delta u_{i,k}$ was also penalized in the reward function.

The reward function was defined in such a way that it was always negative:

$$r(\boldsymbol{s}_{i,k}, \widetilde{u}_{i,k}) = \begin{cases} R_{\text{abort}}, \text{ if episode is aborted} \\ -\left(w_e k_e \left|e_{i,k}\right| + w_P k_P \left|P_{i,k}\right| + w_{\Delta u} k_{\Delta u} \left|\Delta u_{i,k}\right|\right), \text{ else} \end{cases}. \tag{19}$$

If the episode was not aborted, the weighted (w) and normalized (k) error $e_{i,k}$, power consumption $P_{i,k}$ and action change $\Delta u_{i,k}$ created the achieved reward. The normalized state components and actions were used.

Several conditions led to a premature end of an episode and were related to safety and error constraints. To not crash into the preceding vehicle, the distance must always be larger than 0 m. Additionally, we chose an upper distance limit of 50 m:

$$0 \text{ m} < d_{i,k} < 50 \text{ m}. \tag{20}$$

At the same time, this limited the maximum platoon length, thereby increasing the road capacity. To increase safety, the relative velocity also was limited:

$$-5 \text{ m/s} < v_{i-1,k} - v_{i,k} < 5 \text{ m/s}. \tag{21}$$

If the episode was aborted, a negative reward R_{abort} was assigned. The trained state-value function V_π contained the information on whether specific states most likely were going to lead to a premature end of an episode (leading to a low return). Via the estimated advantage $\hat{A}_{k,\boldsymbol{\theta}}$, this was learned by the agent during training (cf. (11)). This meant that for example the agent already attempted to avoid distances $d_{i,k}$ larger yet close to 0 m.

Maximizing the reward function yielded the desired behavior: error minimization, minimization of the power consumption, and an enhanced ride comfort by providing a smooth target acceleration signal. Two different sets of reward weights were heuristically determined and are listed in Table 3. The error-minimizing weight set (EM-RL) focused on minimizing the error $e_{i,k}$, while the power-minimizing weight set (PM-RL) also explicitly considered the minimization of the absolute power consumption $\left|P_{i,k}\right|$. The minimization of $\left|P_{i,k}\right|$ aimed at reducing the consumed energy E_i at the end of the episode τ_i: $E_i = \int_0^T P_i(t)dt$. Even though recuperation was possible, every acceleration and deceleration led to additional losses. Therefore, the less acceleration and deceleration was performed, the less powertrain losses occurred (smaller $\left|P_{i,k}\right|$), which yielded a smaller energy consumption E_i.

Table 3. Reward function parametrizations: the error-minimizing reward parametrization (EM-RL) and power-minimizing reward parametrization (PM-RL).

	w_e	w_P	$w_{\Delta u}$	R_{abort}
EM-RL	1.0	0	0.1	−1000
PM-RL	0.5	6.0	0.1	−100,000

In case of PM-RL, the weight w_e for the error penalization was reduced to enable the agent to focus less on the error $e_{i,k}$, and thereby to deliberately deviate from the set point distance $d^*_{i,k}$ to reduce E_i. In the case of EM-RL, the weight w_P was set to zero, which was why the power was not explicitly considered.

4. Simulative Assessment

PPO was deployed with a policy network of two hidden layers with 64 neurons, each with the $\tanh(\cdot)$ activation function and using the Adam optimizer [49]. The default hyperparameters [45] were used, and $n_{\mathcal{P}} = 4$ parallel processes $\mathcal{P}$ were utilized to achieve a lower computational time of the training. The minibatch scaling factor was increased from the default value $n_{\text{mini}} = 4$ to $n_{\text{mini}} = 16$ to create smaller minibatches and thereby further reduce the computational time (cf. Algorithm 1 and its description in Section 3.1). Since the default hyperparameters provided good training results, no further effort to optimize the hyperparameters was spent.

Training with the EM-RL reward function was performed with a batch size of 4·128 steps ($T_{\mathcal{P}} = 128$ steps per process were used as defined as default in [45]) and a training length of 2×10^6 time steps. Training with the PM-RL reward function was more complicated due to the added control goal of power minimization. This is why the batch size is increased to 4·256 steps ($T_{\mathcal{P}} = 256$) to stabilize the training. On the one hand, this led to a lowered sample efficiency, as more samples needed to be assessed before the policy is updated. On the other hand, the information from more samples was averaged before an update, leading to the policy being less prone to converge to local optima. As this also slowed the training process, the training length was increased to 4×10^6 time steps. See Appendix C for information on average run times.

The training outcome changed based on the initial random seeds (i.e., numbers used to initialize pseudorandom number generators), which affected the randomly assigned initial neural network weights, as well as the environment initialization. A reproducible training setup is a known issue in DRL [50]; to this end, several seeds should be used in each training. The authors of [50] suggested the use of at least five seeds; in this work, nine differently seeded runs were performed for each reward parametrization and communication parametrization. For analyzing the training, the mean return and the standard error were calculated in bins of 50,000 time steps for all training runs.

The DRL environment was created in the Python-based DRL framework OpenAI Gym [51], which provided standardized interfaces to connect DRL environments with DRL algorithms. The Modelica model of the vehicle was exported as a functional-mock-up unit (FMU) [52]. The software package PySimulator [53,54] was used to connect to the FMU and to create a Python-based interface, which could then be directly addressed by the DRL environment. This toolchain was first applied by our group in [9].

4.1. Training

Each CACC environment was initialized with a processed NGSIM velocity and acceleration trajectory. The trajectories were sampled from the NGSIM training set. A training episode had a length of t_T (cf. Table 4), if it was not aborted earlier.

Table 4. Training setup and initialization.

Description	Parameter	Value or Sampling Interval
Episode length	t_T	120 s
Distance at standstill	$r_{c,2}$	2.0 m
Time headway	$t_{h,2}$	0.74 s
Initial acceleration	$a_{2,0}$	0 m/s^2
Initial velocity deviation	Δv_{rand}	[−2.5 m/s, 2.5 m/s)
Initial leader vehicle position	$x_{2,0}$	0 m
Initial distance deviation to leader vehicle	Δd_{rand}	[−10 m, 10 m)
Actions from prior two time steps	$u_{2,-1}$, $u_{2,-2}$	0 m/s^2

The initial velocity of the follower vehicle was selected as

$$v_{2,0} = v_{1,0} + \Delta v_{\text{rand}}, \tag{22}$$

where $v_{2,0}$ was limited to be not below 0 m/s.

With $v_{2,0}$, $d^*_{2,0}$ can be calculated according to (1). The initial distance $d_{2,0}$ is defined to be equal to $d^*_{2,0}$ plus a value Δd_{rand}:

$$d_{2,0} = d^*_{2,0} + \Delta d_{\text{rand}}. \tag{23}$$

$d_{2,0}$ was limited to be not below $r_{c,2}$. The initial values, as well as the sampling intervals (sampling performed with continuous uniform distribution) of Δv_{rand} and Δd_{rand} can be found in Table 4.

The random initialization of $v_{2,0}$ and $d_{2,0}$ as well as the random sampling of a leader vehicle trajectory, aimed at forcing the agent to cope with a wide variety of initial states and the avoidance of overfitting. Randomly initializing the environment helped the agent to explore the state space, which was inherently considered by the algorithm by using a stochastic policy during training.

The training results for different prediction horizons N_k, training communication qualities, and reward functions are shown in Figure 4.

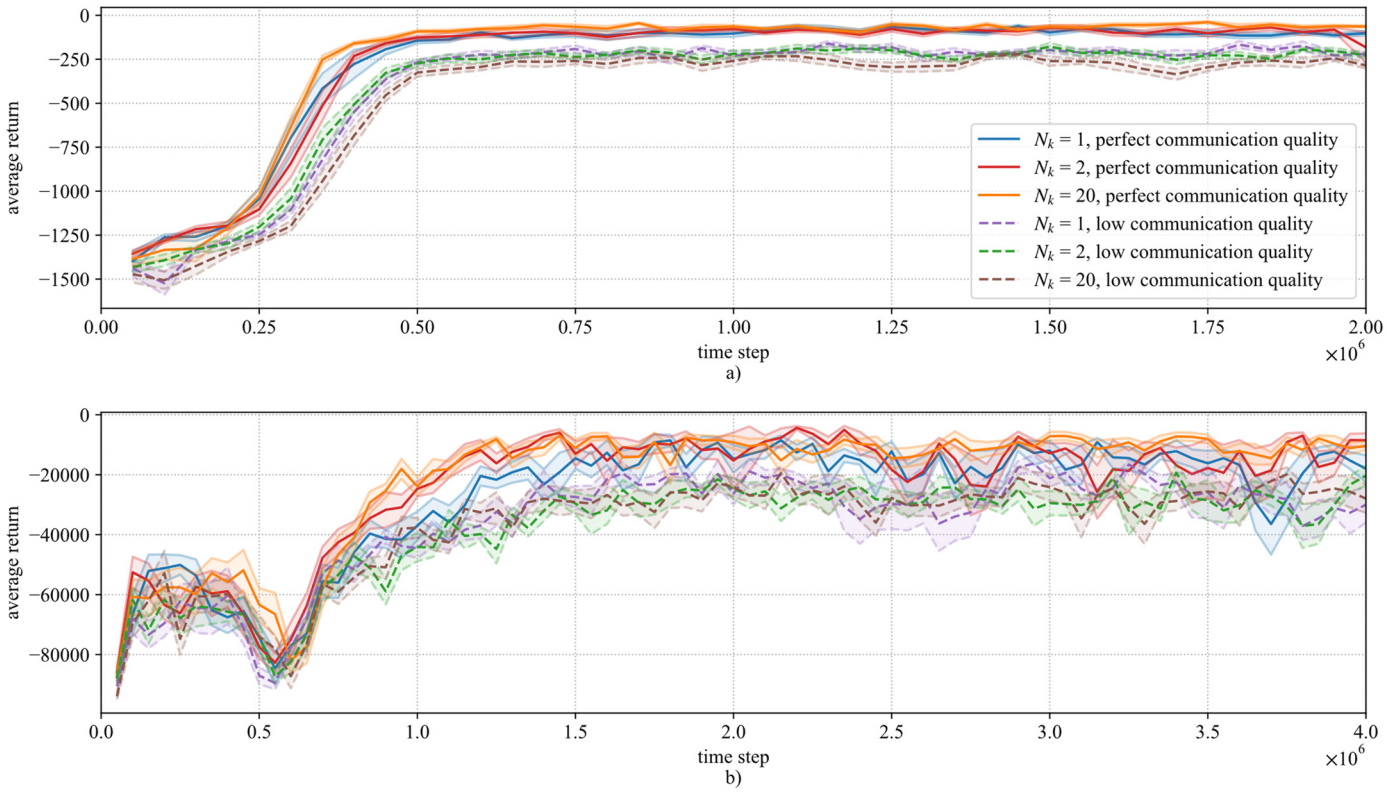

Figure 4. Average return and standard error for multiple runs of the EM-RL (**a**) and PM-RL (**b**) reward function parametrization.

The training evolution with the EM-RL reward function is depicted in Figure 4a. One can see a stable training behavior, with a low standard error for all trainings. The average return converged at approximately 0.5×10^6 time steps. Initially, the average return was below $R_{\text{abort}} = -1000$, because the agents explored different policies, and many lead to an episode being aborted. Comparing the curves for perfect communication quality with prediction horizons $N_k = 1$, 2 and 20, one can see that there was a slight increase in sample efficiency (i.e., convergence speed w.r.t. time steps) for higher prediction horizons, and all prediction horizons showed good training stability. This suggested that the agent could make use of the additional information provided by an increased preview.

The training with low communication quality showed a drop in average return, and the training performance was worse regarding the achieved final average return. Since the

agent only had access to less reliable information, the achievable performance was reduced. This could also be observed for PM-RL.

The PM-RL weight set led to higher absolute rewards due to the nonzero weight w_P compared to the EM-RL weight set, as can be seen in Figure 4b. Additionally, it showed a less stable training behavior. Therefore, R_{abort} was increased (cf. Table 3) to ensure that the agent focused on the safety and error constraint (cf. (20) and (21)).

The average return rose during the first 0.2×10^6 to 0.5×10^6 time steps. After reaching a peak, the return dropped and rose again shortly after until convergence. This can be explained by the opponent reward terms regarding the error $e_{i,k}$ and the power consumption $P_{i,k}$: by following the leader vehicle but accelerating with lower absolute amplitude than it, the agent could easily decrease $P_{i,k}$. However, at the same time, this increased $e_{i,k}$. Only by exploration and considering the possible air drag reduction could a policy be found that reduced both $P_{i,k}$ and $e_{i,k}$.

The general occurrence of training instability of PM-RL compared to EM-RL might have been created by including both the power consumption $P_{i,k}$ and the change in control value $\Delta u_{i,k}$ in the reward function: The penalized power consumption $\left|P_{i,k}\right|$ could also reward control value smoothness, even though this was not its actual aim. Therefore, this created an ambiguous goal for the optimization.

When comparing the PM-RL curves for perfect training communication quality with prediction horizons $N_k = 1$, $N_k = 2$, and $N_k = 20$, one can see that there was an increase in sample efficiency after 0.7×10^6 time steps for $N_k = 2$ and $N_k = 20$. This showed that the additional information acquired through the prediction horizon increased the training performance.

4.2. Test Set Evaluation

For validating the performance of the EM-RL and the PM-RL agents, they were compared to a model-based proportional-derivative (PD) controller, extended with a feedforward (FF) element to incorporate transmitted acceleration information [26] (cf. Appendix A for more information). This controller was referred to as a model-based PDFF (mb-PDFF) controller, and was applied to the test set with both reward function parametrizations. It was also implemented in Modelica/Dymola, like the vehicle model.

Because the design of the mb-PDFF from [26] did not consider any preview; i.e., only the present acceleration was transmitted, the mb-PDFF was exclusively compared with DRL results with a prediction horizon of $N_k = 1$.

For evaluating the performance of the trained policies, they were deterministically applied on the test set. For this, the random initialization was turned off; i.e., $\Delta v_{\text{rand}} = 0$ and $\Delta d_{\text{rand}} = 0$ (cf. (22) and (23)).

It was necessary to select a specific agent out of the set of trained agents to make a direct comparison, as only a single agent could be used as controller in the follower vehicle. The policies with the least amount of crashes in the test set were selected. For training with perfect communication quality, only the test set results with perfect communication quality were considered. For training with low communication quality, the test set results with both perfect and low communication were considered, because the trained agent was expected to perform well in both situations. In case of the same number of crashes, the agent with a higher mean return in the test set $\overline{G}_{\text{test}}$ was selected. $\overline{G}_{\text{test}}$ is defined as $\overline{G}_{\text{test}} = 1/N_{\tau,\text{valid}} \sum_{k=1}^{N_{\tau,\text{valid}}} G_k$, with $N_{\tau,\text{ valid}}$ being the number of episodes that were not aborted in the evaluation of the test.

Table 5 shows the policies corresponding to the best performing runs for each of the reward function parametrizations and communication qualities. The shown DRL

evaluation results were obtained with perfect communication quality. In addition to the number of test set aborts, the mean energy consumption:

$$\overline{E}_{\text{test}} = 1/N_{\tau,\text{valid}} \sum_{k=1}^{N_{\tau,\text{valid}}} E_k \tag{24}$$

and the root mean squared error:

$$RMSE_{\text{test}} = \left(1/N_{\text{s,valid}} \sum_{k=1}^{N_{\text{s,valid}}} e_{2,k}^2 \right)^{0.5} \tag{25}$$

were evaluated. The $RMSE_{\text{test}}$ was calculated with respect to the episode steps $N_{\text{s,valid}}$ that corresponded to $N_{\tau,\text{valid}}$.

Table 5. mb-PDFF and RL-agent-based CACC performance on test set. Perfect communication quality is available during evaluation and the RL agents use a prediction horizon of N_k = 1. The test set consists of 93 trajectories.

	mb-PDFF	**Perfect Comm. Quality during Training**		**Low Comm. Quality during Training**	
		P-EM-RL	**P-PM-RL**	**L-EM-RL**	**L-PM-RL**
$\boldsymbol{RMSE_{\text{test}}}$ **(m)**	0.2055	0.04383	0.7497	0.05160	0.5424
$\boldsymbol{\overline{E}_{\text{test}}}$ **(Wh)**	34.05	30.69	27.95	31.12	29.51
Aborts	0	1	1	1	1

First, mb-PDFF was compared with EM-RL and PM-RL, each trained with perfect communication quality (P-EM-RL, P-PM-RL).

The P-EM-RL agent showed the best performance with respect to the minimization of e. The P-EM-RL showed a much smaller $RMSE_{\text{test}}$ than mb-PDFF (−78.7%). This can be explained by model uncertainties being considered in the design of the mb-PDFF, which made the mb-PDFF less aggressive in order to be capable of adjusting to parameter deviations. Additionally, the P-EM-RL had a lower $\overline{E}_{\text{test}}$ (−9.9%) than mb-PDFF, because the P-EM-RL was trained to minimize the change in control value.

P-PM-RL was able to reduce $\overline{E}_{\text{test}}$ by 17.9% compared to mb-PDFF and 8.9% compared to P-EM-RL. One can see that the deliberate deviation from the set point distance $d_{2,k}^*$ was used to decrease the power consumption, and thereby decrease the energy consumption. This also led to a significant increase in the distance error: P-PM-RL's $RMSE_{\text{test}}$ was 3.6-fold larger than mb-PDFF's, and 17.1-fold larger than P-EM-RL's.

Subsequently, EM-RL and PM-RL trained with low communication quality (L-EM-RL, L-PM-RL) were considered. Both L-EM-RL and L-PM-RL performed similarly to their counterparts P-EM-RL and P-PM-RL, but slightly worse (L-PM-RL had a lower $RMSE_{\text{test}}$ than P-PM-RL, but a higher $\overline{E}_{\text{test}}$). This can be explained by the occurrences of burst errors during training, which made the agents rely less on the transmitted acceleration information.

This led to L-EM-RL being less capable of reaching the set point distance $d_{2,k}^*$ than P-EM-RL, while simultaneously having an increased $\overline{E}_{\text{test}}$. Similarly, this led to L-PM-RL being more careful with deliberately deviating from $d_{2,k}^*$ than P-PM-RL, and therefore also having an increased $\overline{E}_{\text{test}}$.

For all DRL agents, one test set abort occurred out of the 93 tested trajectories. This was related to the strict string-stability implementation of the DRL agents (as opposed to only considering string stability during the tuning of the mb-PDFF's parameters, cf. Appendix A) even in emergency breaking situations.

The evaluation results for higher prediction horizons ($N_k = 2$ and $N_k = 20$) are summarized in Appendix B (cf. Table 2; the DRL results in Table 5 are an extract from Table 2).

Larger prediction horizons only partially yielded a better performance of the agents. Only $N_k = 2$ yielded a lower $RMSE_{\text{test}}$ for P-EM-RL, while neither $N_k = 2$ nor $N_k = 20$ improved L-EM-RL's performance. This can be explained by the set point distance $d^*_{2,k}$ tracking problem to be sufficiently solvable with $N_k = 2$. At the same time, an increased prediction horizon meant for the agent to identify the superfluous information, and therefore increased the difficulty. This was especially true for low communication quality, for which correct acceleration information was partially not available.

For P-PM-RL, a prediction horizon increase from $N_k = 1$ to $N_k = 2$ led to a lower $RMSE_{\text{test}}$, while $\overline{E}_{\text{test}}$ stayed approximately the same. A further horizon increase from $N_k = 1$ to $N_k = 20$ led to a lower $\overline{E}_{\text{test}}$ and $RMSE_{\text{test}}$ (w.r.t. perfect communication quality during evaluation). For L-PM-RL, an increased prediction horizon led to a lower $\overline{E}_{\text{test}}$, but a higher $RMSE_{\text{test}}$. This showed that even for low communication quality, the energy minimization objective benefited from a larger prediction horizon.

Overall, training with low communication quality resulted in a policy that was more robust regarding $\overline{E}_{\text{test}}$ and $RMSE_{\text{test}}$ during low communication quality. However, the agents trained with low communication quality also showed multiple test set aborts during evaluation with low communication quality.

4.3. Controller Analysis

In the following section, the mb-PDFF, P-EM-RL, and P-PM-RL (the latter two with a prediction horizon of N_k = 1) are analyzed in time domain with one trajectory chosen for the leader vehicle.

Figure 5a shows the velocity of the leader vehicle, as well as the velocity of the follower vehicle, as it was created by the mb-PDFF, the P-EM-RL agent, and the P-PM-RL agent. The good performance regarding the distance error minimization of P-EM-RL is clearly visible in Figure 5b.

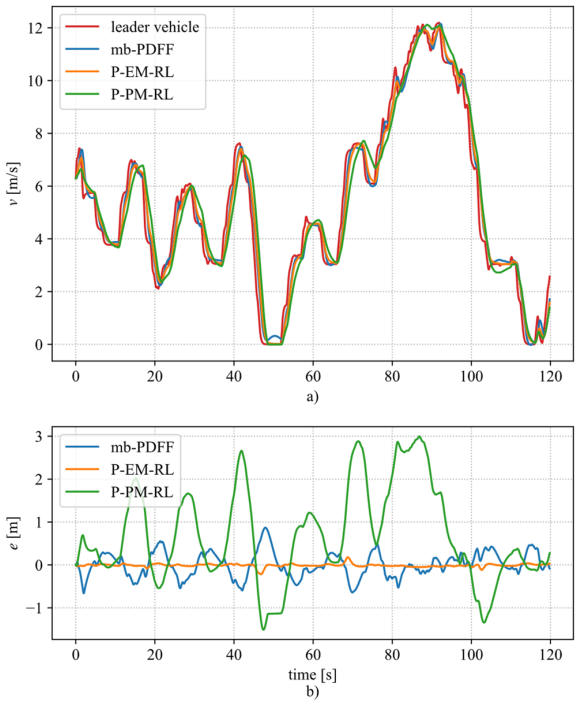

Figure 5. Velocity v of the leader vehicle, as well as the follower vehicle (**a**). The velocity of the latter was created by mb-PDFF, P-EM-RL, and P-PM-RL; the respective distance errors $e = e_{2,k}$ are shown in (**b**).

It can be seen in Figure 5a that the follower vehicle mimicked the driving behavior of the leader vehicle. P-PM-RL showed undershoots after longer periods of deceleration (e.g., at 105 s), which resulted from deliberately deviating from the target distance $d^*_{2,k}$ (cf. Figure 5b). Before the undershoot, P-PM-RL drove faster than the leader vehicle. At the end of the deceleration phase, the follower vehicle drove closer to the preceding vehicle to benefit from the reduced air drag. Afterwards, the reduced distance was compensated by driving slower than the leader vehicle, which meant by performing an undershoot. It should be noted that these undershoots resulted in negative velocity in the case of the preceding vehicle's velocity approaching 0 m/s (e.g., Figure 5a at 50 s). This behavior was possible during training and driving cycle set evaluation, but was forbidden here in the time domain analysis.

Figure 6 shows the control value $u = u_{2,k}$ for mb-PDFF, P-EM-RL, and P-PM-RL for the time interval of 60 s to 80 s of the episode (the time interval was well suited to show the characteristic controllers' behavior and the effect of the string-stability limit). It can be seen that the amplitude of P-EM-RL was lower than that of the mb-PDFF, which most likely was caused by the penalization of the change in control value $\Delta u_{i,k}$ in its reward function. This showed that the P-EM-RL was less aggressive in general than the mb-PDFF (leading to a lower energy consumption), while having a better error-minimizing performance (cf. Table 5 and Figure 5b).

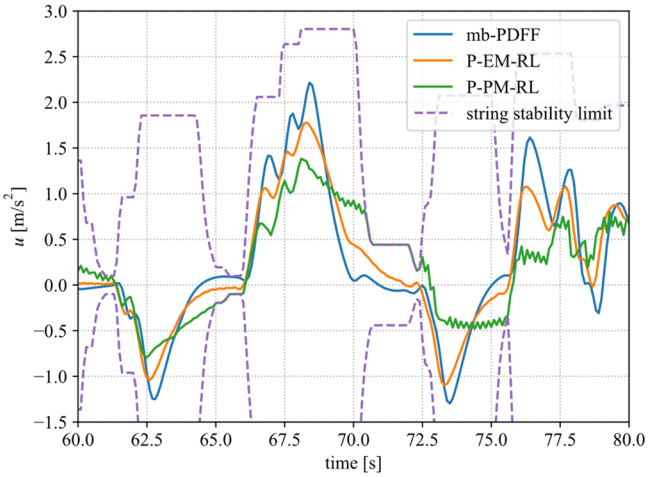

Figure 6. Control value $u = u_{2,k}$ with respective string-stability limit, created by mb-PDFF, P-EM-RL, and P-PM-RL. The time interval from 60 s to 80 s of the simulation with a total length of 120 s is shown.

Regarding the constraints, the figure shows that the control value of the mb-PDFF and both agents remained within a modest range (cf. (14)), that the DRL agents respected the string stability limit, and that, usually, the mb-PDFF also was compatible with the DRL agents' string-stability limit.

The P-PM-RL agent still shows a nonsmooth signal due to the ambiguity created by penalizing both the power consumption and the change in control value. During periods of acceleration and braking, it showed lower maximum absolute accelerations, but higher absolute accelerations at the end of the periods than mb-PDFF and P-EM-RL. This allowed it to deviate from the target distance $d^*_{2,k}$.

For analyzing the trained agents' string-stability property, a platoon with three P-EM-RL-controlled follower vehicles was simulated (Figure 7a,b) and three P-PM-RL-controlled follower vehicles (Figure 7c,d) (P-EM-RL and P-PM-RL both with prediction horizon of $N_k = 1$).

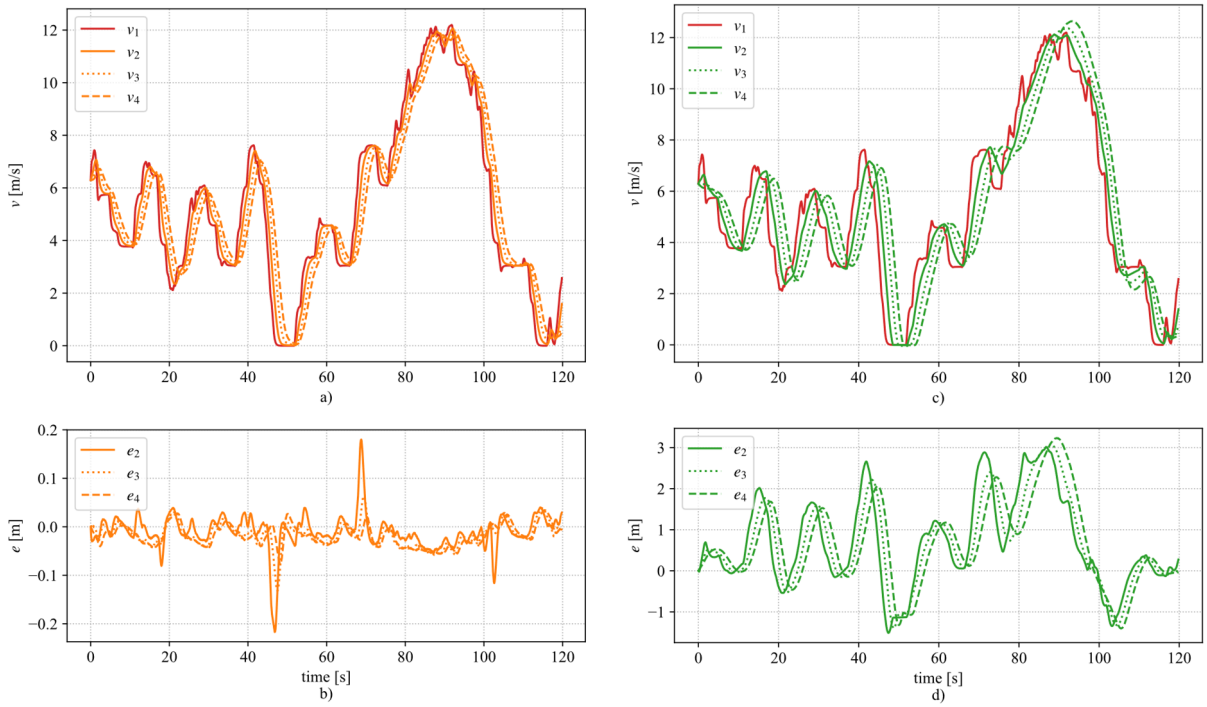

Figure 7. Platoon consisting of three agent-controlled vehicles and a leader vehicle. (**a**,**b**) show the velocities v and the respective distance errors e in case the follower vehicles were P-EM-RL-controlled; (**c**,**d**) show v and e in case the follower vehicles were P-PM-RL-controlled.

The P-EM-RL-controlled follower vehicles behaved as string stable: each follower vehicle followed the preceding vehicle without an acceleration amplification downstream. This also led to a reduction of the maximum absolute distance error values, as can be seen in Figure 7b at 50 s, 70 s, and 105 s.

The P-PM-RL-controlled vehicles mostly behaved as string stable, but also showed instabilities. The velocity over- and undershoots with respect to the preceding vehicle were amplified downstream at 90 s and 105 s, such that the absolute distance error also was amplified. For negative distance errors, this may lead to a crash. This was not fully prevented by the string stability limit (cf. (16)), because it only referred to the maximum absolute acceleration value of the preceding vehicle within the string stability time window N_{SS}, but not the duration of this maximum acceleration. Thus, in future work, the string-stability limit also needs to restrict the acceleration duration of the follower vehicle. Additionally, this reduction of possible acceleration amplitude would increase the passenger comfort.

The string stability behavior of the P-EM-RL-controlled vehicles also suggested the assessment of P-EM-RL-controllers in platoons with more than three agent-controlled vehicles, whereas the P-PM-RL-controllers first required a stricter string-stability limit.

5. Discussion and Outlook

This work assessed the use of deep reinforcement learning for CACC.

A buffer was used in the C2C communication to store the predicted acceleration information. This enabled the use of a prediction horizon of multiple future time steps. The occurrence of burst errors was modeled with a Markov chain, and burst errors were marked as invalid information in the buffer. Furthermore, a string-stability condition was incorporated by limiting the acceleration set point of the follower vehicle with respect to future accelerations of the preceding vehicle. Thereby, accelerations could not be amplified downstream in the platoon without limit.

If the training and application of deep reinforcement learning is not properly designed, robustness and overfitting problems might occur; i.e., the learning-based algorithm is

highly optimized for the data synthetically generated during simulation, failing to cope with data generated by the real system. This is particularly worrisome when transferring the algorithm from a simulation environment to the real world, leading to potential loss of performance (best scenario) and safety (worst case). To mitigate these issues in our work, model randomization techniques were incorporated into the learning process. The key idea was to inject parameter variability in the simulation model, exposing the agent to model inaccuracies during the learning phase, yielding robust control policies. In our work, particular attention was dedicated to exposing the control algorithm to different levels of communication delays and errors, one of the main sources of uncertainty in the platooning applications.

Multiple agents were trained and evaluated on an extra test set that was not used for training. The best performing agents were validated by comparison with a model-based PD controller with feedforward. Here, the distance error minimizing CACC agent was able to outperform the model-based controller by reducing the $RMSE_{\text{test}}$ by up to 78.7% on the test set.

In comparison with the model-based controller, the energy minimizing CACC agent reduced the mean energy consumption by 17.9% on the test set. For the error-minimizing agents, it was shown that only small increases of the prediction horizon length yielded a better error minimization, whereas the energy consumption could also be improved by large increases of the prediction horizon length. Moreover, it was shown that considering burst errors during training created an agent that was robust against them and performed well regarding distance errors and energy minimization.

All in all, it was possible to train agents that could: (1) minimize their energy consumption; (2) provide string stability to a given extent; (3) exploit preview information available in the communication; and (4) minimize the effect of burst errors.

However, to guarantee string stability more generally, the string stability limit should be able to change depending on the driving situation. Most of the time, it should be stricter, and also should consider the acceleration duration of the preceding vehicle instead of only the maximum acceleration. During test set evaluation of the DRL agents, some test set aborts occurred due to the string-stability limit in emergency breaking situations. Thus, the string-stability limit should also be less restrictive when the distance to the preceding vehicle is very low.

The experienced predisposition of DRL towards unsmooth control signals is a disadvantage, and must be considered when transferring the controller to real-world applications. Unsmooth acceleration control signals lead to high jerks and reduce driving comfort. In addition, they increase energy consumption.

In the future, we plan to validate the obtained results with experiments on real-word vehicles.

Author Contributions: Conceptualization, R.d.C., J.M., J.U. and J.B.; methodology, J.M., R.d.C. and J.U.; software, J.M., J.U., R.d.C. and J.B.; RL and FMI framework, J.U. and J.M.; validation, J.M. and R.d.C.; writing—original draft preparation, J.M.; writing—review and editing, J.M., R.d.C., J.U. and J.B.; visualization, J.M., J.U. and R.d.C.; supervision, R.d.C. and J.B. All authors have read and agreed to the published version of the manuscript.

Funding: The work of J.M., J.B. and J.U was funded by the DLR internal project NGC-KoFiF & Intelligent Mobility. These projects also funded R.d.C. during his employment at DLR.

Data Availability Statement: Not applicable.

Acknowledgments: The authors' thanks go to Christina Schreppel for her valuable support.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Model-Based PD Controller with Feedforward

The mb-PDFF was fast due to the proportional element, and featured a derivative component to increase the controller's stability. Because of fast-changing set points in

trajectory following, there was no need for an integral element. The control signal u_i for platoon vehicle i was composed partially of a feedback signal $u_{PD,i}$ and a feedforward signal $u_{ff,i}$:

$$u_i = u_{PD,i} + u_{ff,i}. \tag{A1}$$

$u_{PD,i}$ is calculated as:

$$u_{PD,i}(j\omega) = k_{p,i} \cdot e_i(j\omega) + j\omega \cdot k_{d,i} \cdot e_i(j\omega), \tag{A2}$$

where e_i again is the distance error (cf. (2)). The feedforward signal $u_{ff,i}$ consists of a filter block and a block that models the communication delay Δ_i (w.r.t. the transmitted present acceleration of the preceding vehicle). The filter was selected such that the disturbance effect of the preceding vehicle's position $x_{i-1}(t)$ was reduced, using a specific value for the time headway $t_{h,i}$ (cf. [26] and Table 4).

According to [26], the controller's parameters $t_{h,i}$, $k_{p,i}$, and $k_{d,i}$ were determined with respect to specifications of pole placement, string stability, and nonparametric variations. The parameters were calculated with the parameter space method [55], and can be found in Table A1.

Table A1. Model-based PDFF controller parametrization.

Parameter	Value
$k_{p,i}$	0.49
$k_{d,i}$	0.70

Both the DRL-based CACCs and the mb-PDFF used the same distance at standstill $r_{c,2}$, the same time headway $t_{h,2}$, and the same delay Δ_2 (cf. Tables 1 and 4).

Appendix B. Full Test Set Evaluation Results for Agents

Table 2 summarizes the driving cycle test set evaluation results for the EM-RL and PM-RL agents for prediction horizons $N_k = 1$, 2 and 20, as well as for perfect and low communication quality during training.

Table 2. Results of trained agents for the driving cycle test set during both perfect and low communication quality. The test set consisted of 93 trajectories.

		Perfect Communication Quality during Training						Low Communication Quality during Training					
		P-EM-RL			P-PM-RL			L-EM-RL			L-PM-RL		
		N_k = 1	N_k = 2	N_k = 20	N_k = 1	N_k = 2	N_k = 20	N_k = 1	N_k = 2	N_k = 20	N_k = 1	N_k = 2	N_k = 20
Perfect Comm. Quality during Evaluation	$RMSE_{\text{test}}$ (m)	0.0438	0.0403	0.158	0.750	0.370	0.624	0.0516	0.127	0.100	0.542	0.733	0.803
	$\overline{E}_{\text{test}}$ (Wh)	30.7	30.0	28.9	28.0	28.2	24.6	31.1	29.9	28.4	29.5	28.4	23.9
	Aborts	1	1	0	1	0	2	1	0	1	1	1	0
Low Comm. Quality during Evaluation	$RMSE_{\text{test}}$ (m)	0.731	0.487	0.593	3.10	1.49	4.91	0.269	0.317	0.392	0.535	0.680	0.874
	$\overline{E}_{\text{test}}$ (Wh)	37.5	36.5	34.0	35.9	36.2	35.7	31.4	30.5	30.7	30.0	29.3	25.3
	Aborts	8	9	5	6	6	30	9	7	8	9	11	4

Appendix C. Required Computational Time and Hardware Setup

In this work, an Intel Xeon W-2135 processor was used (3.7 GHz, 6 cores, 12 threads) together with 32 GB RAM and Tensorflow 1.12.0. Table A3 gives an overview of the required average run times for the different prediction horizons N_k of the EM-RL training with 2×10^6 time steps and the PM-RL training with 4×10^6 time steps, each with perfect communication quality. For low communication quality, the average run times were similar.

Table A3. Average run times of EM-RL and PM-RL training runs with perfect communication quality.

Reward Function Parametrization	Prediction Horizon N_k	Average Run Time (s)
EM-RL	1	7390
	2	7960
	20	11,600
PM-RL	1	12,800
	2	14,100
	20	25,000

References

1. Jia, D.; Lu, K.; Wang, J.; Zhang, X.; Shen, X. A Survey on Platoon-Based Vehicular Cyber-Physical Systems. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 263–284. [CrossRef]
2. Turri, V.; Besselink, B.; Martensson, J.; Johansson, K. Fuel-Efficient Heavy-Duty Vehicle Platooning by Look-Ahead Control. In Proceedings of the 53rd IEEE Conference on Decision and Control, Los Angeles, CA, USA, 15–17 December 2014.
3. Guanetti, J.; Kim, Y.; Borrelli, F. Control of Connected and Automated Vehicles: State of the Art and Future Challenges. *Annu. Rev. Control* **2018**, *45*, 18–40. [CrossRef]
4. Seiler, P.; Pant, A.; Hedrick, K. Disturbance Propagation in Vehicle Strings. *IEEE Trans. Autom. Control* **2004**, *49*, 1835–1842. [CrossRef]
5. Sutton, R.; Barto, A. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA; London, UK, 2018.
6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. Available online: https://arxiv.org/abs/1312.5602 (accessed on 19 December 2013).
7. Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **2016**, *529*, 484–489. [CrossRef]
8. OpenAI; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680. Available online: https://arxiv.org/abs/1912.06680 (accessed on 13 December 2019).
9. Ultsch, J.; Brembeck, J.; de Castro, R. Learning-Based Path Following Control for an Over-Actuated Robotic Vehicle. In Proceedings of the AUTOREG 2019, Mannheim, Germany, 2–3 July 2019.
10. Duan, J.; Shi, D.; Diao, R.; Li, H.; Wang, Z.; Zhang, B.; Bian, D.; Yi, Z. Deep-Reinforcement-Learning-Based Autonomous Voltage Control for Power Grid Operations. *IEEE Trans. Power Syst.* **2019**, *35*, 814–817. [CrossRef]
11. Rabault, J.; Kuchta, M.; Jensen, A.; Réglade, U.; Cerardi, N. Artificial Neural Networks Trained through Deep Reinforcement Learning Discover Control Strategies for Active Flow Control. *J. Fluid Mech.* **2019**, *865*, 281–302. [CrossRef]
12. Chu, T.; Wang, J.; Codecà, L.; Li, Z. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 1086–1095. [CrossRef]
13. de Castro, R.; Schaub, A.; Satzger, C.; Brembeck, J. A Vehicle Following Controller for Highly-Actuated Vehicles. In Proceedings of the 13th International Symposium on Advanced Vehicle Control, Munich, Germany, 13–16 September 2016.
14. Wu, C.; Lin, Y.; Eskandarian, A. Cooperative Adaptive Cruise Control With Adaptive Kalman Filter Subject to Temporary Communication Loss. *IEEE Access* **2019**, *7*, 93558–93568. [CrossRef]
15. Dunbar, W.; Caveney, D. Distributed Receding Horizon Control of Vehicle Platoons: Stability and String Stability. *IEEE Trans. Autom. Control* **2012**, *57*, 620–633. [CrossRef]
16. Buechel, M.; Knoll, A. Deep Reinforcement Learning for Predictive Longitudinal Control of Automated Vehicles. In Proceedings of the 21st International Conference on Intelligent Transportation Systems, Maui, HI, USA, 4–7 November 2018.
17. Everett, M.; Lütjens, B.; How, J. Certifiable Robustness to Adversarial State Uncertainty in Deep Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst. (Early Access)* **2021**, 1–15. [CrossRef]
18. Ultsch, J.; Mirwald, J.; Brembeck, J.; de Castro, R. Reinforcement Learning-based Path Following Control for a Vehicle with Variable Delay in the Drivetrain. In Proceedings of the IEEE Intelligent Vehicles Symposium, Las Vegas, NV, USA, 19 October–13 November 2020.
19. Wei, Z.; Jiang, Y.; Liao, X.; Qi, X.; Wang, Z.; Wu, G.; Hao, P.; Barth, M. End-to-End Vision-Based Adaptive Cruise Control (ACC) Using Deep Reinforcement Learning. *arXiv* **2001**, arXiv:2001.09181. Available online: https://arxiv.org/abs/2001.09181 (accessed on 24 January 2020).
20. Lin, Y.; McPhee, J.; Azad, N. Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control. *IEEE Trans. Intell. Veh.* **2021**, *6*, 221–231. [CrossRef]
21. Desjardins, C.; Chaib-draa, B. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 1248–1260. [CrossRef]
22. Chu, T.; Kalabić, U. Model-based Deep Reinforcement Learning for CACC in Mixed-Autonomy Vehicle Platoon. In Proceedings of the IEEE 58th Conference on Decision and Control, Nice, France, 11–13 December 2019.

23. Wei, S.; Zou, Y.; Zhang, T.; Zhang, X.; Wang, W. Design and Experimental Validation of a Cooperative Adaptive Cruise Control System Based on Supervised Reinforcement Learning. *Appl. Sci.* **2018**, *8*, 1014. [CrossRef]
24. Peake, A.; McCalmon, J.; Raiford, B.; Liu, T.; Alqahtani, S. Multi-Agent Reinforcement Learning for Cooperative Adaptive Cruise Control. In Proceedings of the IEEE 32nd International Conference on Tools with Artificial Intelligence, Baltimore, MD, USA, 9–11 November 2020.
25. Mirwald, J. Platooning Control with Deep Reinforcement Learning. Master's Thesis, Technical University of Munich, Munich, Germany, 2019.
26. de Castro, R.; Brembeck, J. A Command Governor Approach for Platooning Applications. In Proceedings of the IEEE Intelligent Vehicles Symposium, Los Angeles, NV, USA, 11–14 June 2017.
27. Ploeg, J. Analysis Design of Controllers for Cooperative Automated Driving. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2014.
28. Kianfar, R.; Augusto, B.; Ebadighajari, A.; Hakeem, U.; Nilsson, J.; Raza, A.; Tabar, R.; Irukulapati, N.; Englund, C.; Falcone, P.; et al. Design and Experimental Validation of a Cooperative Driving System in the Grand Cooperative Driving Challenge. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 994–1007. [CrossRef]
29. Elliott, E. Estimates of Error Rates for Codes on Burst-Noise Channels. *Bell Syst. Tech. J.* **1963**, *42*, 1977–1997. [CrossRef]
30. Patel, R.; Haerri, J.; Bonnet, C. Centralized Model Predictive CACC Control Robust to Burst Communication Errors. In Proceedings of the IEEE 88th Vehicular Technology Conference, Chicago, IL, USA, 27–30 August 2018.
31. Modelica Association 61 Ideell Förening: Modelica. Available online: https://modelica.org/ (accessed on 15 September 2021).
32. Dassault Systèmes, S.E. Dymola. Available online: https://www.3ds.com/products-services/catia/products/dymola (accessed on 15 September 2021).
33. Zimmer, D.; van der Linden, F.; Qu, Z. Planar Mechanics 1.4.0. In a Free Modelica Library for Planar Mechanical MULTI-body Systems. Available online: https://github.com/dzimmer/PlanarMechanics/releases/tag/v1.4.0 (accessed on 12 January 2017).
34. Brembeck, J.; Ho, L.; Schaub, A.; Satzger, C.; Tobolar, J.; Bals, J.; Hirzinger, G. ROMO—The Robotic Electric Vehicle. In Proceedings of the IAVSD International Symposium on Dynamics of Vehicle on Roads and Tracks, Manchester, UK, 11–14 August 2011.
35. Brembeck, J. Model Based Energy Management State Estimation for the Robotic Electric Vehicle ROboMObil. Ph.D. Thesis, Technical University of Munich, Munich, Germany, 2018.
36. Bertoni, L.; Guanetti, J.; Basso, M.; Masoero, M.; Cetinkunt, S.; Borrelli, F. An Adaptive Cruise Control for Connected Energy-Saving Electric Vehicles. *IFAC-Pap.* **2017**, *50*, 2359–2364. [CrossRef]
37. Turri, V.; Besselink, B.; Johansson, K. Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning. *IEEE Trans. Control. Syst. Technol.* **2017**, *25*, 12–28. [CrossRef]
38. Ahmed, S.; Bayer, B.; Deußen, N.; Emmelmann, H.-J.; Flegl, H.; Gilhaus, A.; Götz, H.; Großmann, H.; Hoffmann, R.; Hucho, W.-H.; et al. *Aerodynamik des Automobils*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 1999.
39. Tobolar, J.; Otter, M.; Bünte, T. Modelling of Vehicle Powertrains with the Modelica PowerTrain Library. In Proceedings of the Systemanalyse in der Kfz-Antriebstechnik IV, Augsburg, Germany, 6–7 March 2007.
40. Schröder, D. *Elektrische Antriebe—Regelung von Antriebssystemen*, 3rd ed.; Springer: Berlin, Germany, 2009.
41. United States Department of Transportation. Next Generation Simulation (NGSIM). Available online: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm (accessed on 31 December 2006).
42. Montanino, M.; Punzo, V. Making NGSIM Data Usable for Studies on Traffic Flow Theory: Multistep Method for Vehicle Trajectory Reconstruction. *Transp. Res. Rec. J. Transp. Res. Board* **2013**, *2390*, 99–111. [CrossRef]
43. Szepesvári, C. Algorithms for Reinforcement Learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*; Morgan & Claypool: San Rafael, CA, USA, 2010.
44. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347v2. Available online: https://arxiv.org/abs/1707.06347v2 (accessed on 28 August 2017).
45. Hill, A.; Raffin, A.; Ernestus, M.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; et al. Stable Baselines—PPO2. Available online: https://github.com/hill-a/stable-baselines; https://stable-baselines.readthedocs.io/en/v2.4.0/modules/ppo2.html (accessed on 17 January 2019).
46. Bishop, C. *Pattern Recognition and Machine Learning*, 8th ed.; Springer: New York, NY, USA, 2009.
47. Heess, N.; TB, D.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S.; et al. Emergence of Locomotion Behaviours in Rich Environments. *arXiv* **2017**, arXiv:1707.02286. Available online: https://arxiv.org/abs/1707.02286 (accessed on 7 July 2017).
48. Raffin, A.; Sokolkov, R. Learning to Drive Smoothly in Minutes. Available online: https://github.com/araffin/learning-to-drive-in-5-minutes/ (accessed on 31 January 2019).
49. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980. Available online: https://arxiv.org/abs/1412.6980 (accessed on 22 December 2014).
50. Islam, R.; Henderson, P.; Gomrokchi, M.; Precup, D. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *arXiv* **2017**, arXiv:1708.04133.
51. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540. Available online: https://arxiv.org/abs/1606.01540 (accessed on 5 June 2016).

52. Modelica Association 61 Ideell Förening: FMI Version 2.0. Available online: https://fmi-standard.org/downloads/ (accessed on 15 September 2021).
53. Ganeson, A.; Fritzson, P.; Rogovchenko, O.; Asghar, A.; Sjölund, M.; Pfeiffer, A. An OpenModelica Python Interface and its use in PySimulator. In Proceedings of the 9th International Modelica Conference, Munich, Germany, 3–5 September 2012.
54. Pfeiffer, A.; Hellerer, M.; Hartweg, S.; Otter, M.; Reiner, M. PySimulator—A Simulation and Analysis Environment in Python with Plugin Infrastructure. In Proceedings of the 9th International Modelica Conference, Munich, Germany, 3–5 September 2012.
55. Ackermann, J. *Robust Control: The Parameter Space Approach*, 2nd ed.; Springer: London, UK, 2002.