electronics

MDPI

Article

Adaptive Control Strategies for Networked Systems: A Reinforcement Learning-Based Approach

André Gilerson [1,*], Niklas Bünte [1], Pierre E. Kehl [1] and Robert H. Schmitt [1,2]

[1] Fraunhofer Institute for Production Technology IPT, 52074 Aachen, Germany
[2] Laboratory for Machine Tools and Production Engineering (WZL), Department Intelligence in Quality Sensing, RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany
* Correspondence: andre.gilerson@ipt.fraunhofer.de; Tel.: +49-241-8904230

Abstract: Advances in industrial 5G communication technologies and robotics create new possibilities while also increasing the complexity and variability of networked control systems. The additional throughput and lower latency provided by 5G networks enable applications such as teleoperation of machinery, flexible reconfigurable robotic manufacturing cells, or automated guided vehicles. These use cases are set up in dynamic network environments where communication latency and jitter become critical factors that must be managed. Despite the advancements in 5G technologies, such as ultra-reliable low-latency communication (URLLC), adaptive control strategies such as reinforcement learning (RL) remain critical to handle unpredictable network conditions and ensure optimal system performance in real-world industrial applications. In this paper, we investigate the potential of RL in scenarios with communication latency similar to a public 5G deployment. Our study includes an incremental improvement by utilizing long short-term memory-based neural networks in combination with proximal policy optimization in this scenario. Our findings indicate that incorporating latency into the training environment enhances the robustness and efficiency of RL controllers, especially in scenarios characterized by variable network delays. This exploration provides insights into the feasibility of using RL for networked control systems and underscores the importance of incorporating realistic network conditions into the training phase.

Keywords: reinforcement learning; artificial intelligence (AI); 5G; networked control systems; network simulation

Academic Editor: Djuradj Budimir

Received: 13 February 2025
Revised: 16 March 2025
Accepted: 20 March 2025
Published: 26 March 2025

Citation: Gilerson, A.; Bünte, N.; Kehl, P.E.; Schmitt, R.H. Adaptive Control Strategies for Networked Systems: A Reinforcement Learning-Based Approach. *Electronics* **2025**, *14*, 1312. https://doi.org/10.3390/electronics14071312

1. Introduction

Advancements in both industrial 5G communication and robotics are opening new frontiers for networked control systems, enabling applications such as teleoperation of machinery, flexible reconfigurable robotic manufacturing cells, and automated guided vehicles [1]. In contrast to traditional control systems, where the plant and controller have a direct, reliable connection that allows for fixed cycle times, networked control systems are connected through a communication network such as 5G, which can suffer from varying latency, jitter, and packet loss [2]. Industrial settings using private 5G campus networks can achieve latencies under 10 ms [3]; however, this is highly dependent on, e.g., the location of the end device in relation to the antennae, whether there are obstacles in their line of sight, or traffic/interference generated by other network participants [4].

There are multiple strategies in control theory to deal with time delay in the control loop; small latencies can be accounted for in traditional feedback control using, e.g., the Nyquist criterion [5]. Other approaches utilize, e.g., Smith predictors [6] or model predictive

control [7]. These approaches generally only perform well for a well-defined, previously known, and constant latency distribution and have no faculties for dealing with loss of control feedback due to packet loss. As previously mentioned, however, the latency distribution in specifically wireless networked systems with mobile devices can vary drastically over time, e.g., because the controlled robot moved into a new location, or other network participants entered and disconnected from the network, etc. One could account for this variability by assuming the worst-case scenario; however, this would effectively limit the performance of the control system [8]. Although it is not possible to analytically describe the latency distribution, the simulation of dynamic wireless networks for a given state is a well-understood and researched topic [9].

AI, and more specifically reinforcement learning (RL), is a currently emerging strategy for the control of complex systems [10]. It offers two main advantages: Firstly, it can easily deal with complex input variables (e.g., a video feed), where traditional control theory suffers from the curse of dimensionality [10]. Secondly, RL is not reliant on a prior analytical description of the control system but rather creates an internal representation through interaction with the system or a simulation of it [10].

This paper presents the approach of adding a simulation of network latency into the training environment of the RL algorithm. We specifically examine RL in the context of mobile 5G robotics. Building on previous research in RL with time delays, we extend the setup of agents' neural networks using long short-term memory (LSTM) and evaluate the applicability of RL for dynamically networked control systems. The combination of LSTM cells with proximal policy optimization (PPO) is an incremental improvement on previous approaches of RL in scenarios with communication latency.

The rest of this paper is structured as follows: Section 2 provides an overview of related works and motivates the selection of the reinforcement learning algorithm described in this paper; Section 3 describes the training environment and the setup of the reinforcement learning algorithms; Section 4 describes the results and the performance of the trained agents; Section 5 provides a discussion and analysis of the results; and Section 6 presents a conclusion and suggestions for future work in this domain.

2. Previous Works

The applicability of RL for the control of complex robotics systems has been shown in various previous publications and will, therefore, not be discussed further in this section [10]. Rather, this section will give an overview of previous works specifically related to RL in systems with communication latencies. The central problem addressed in this paper is that traditional RL algorithms assume immediate and reliable feedback, which is not always feasible in real-world networked environments with dynamic latency. Following this, this section presents an overview of previous research on RL in scenarios with latency and the estimation of latency in 5G networks and finishes by motivating the further research presented in this paper.

2.1. Reinforcement Learning with Network Latency

Mahmood et al. and Xie et al. showed that RL agents trained without awareness of delays suffer when applied to real-world robots with latencies, resulting in a drastic reduction in their performance [11,12]. This is because, traditionally, RL training algorithms assume the system to be a Markov decision process (MDP). Although the latency distribution for packets is dependent on, e.g., the location of the controlled system and therefore on the system state, the chance of each packet being lost or delayed is inherently random. The future state of the system might therefore be dependent on the older system state, violating the Markov assumption [13].

Andrychowicz et al. demonstrate that adding random latencies—which, however, stay constant over a training episode—as a domain randomization is insufficient to deal with the issues of varying latencies [14]. Sandha et al. show that this approach can be improved by providing the measured latency as an additional input to the agents' neural network [15].

The Markov assumption can be restored by extending the state description of the Markov process. For example, Chen et al. propose delay-aware trajectory sampling by buffering several previously taken actions [16]. They show that their algorithm outperforms agents that were trained without network latency but only consider a fixed sample time and constant latency [16]. Bouteiller et al. handle variable latencies by proposing to buffer several actions with their latency, allowing the training algorithm to only reward actions that were executed (instead of lost or overwritten by a newer action) [17]. Derman et al. show that state augmentation leads to an exponential increase in computational complexity with increased time delays since the capability of handling delays is directly linked to the length of the observation/action buffer [18]. They present a different approach through a model-based Q-learning algorithm but do not explore how it performs in scenarios with variable dynamic latencies [18].

Yao et al. propose a model-based approach by recording state transitions, actions taken, and latencies during training and using these to train an additional probabilistic model of the environment that describes the likely state transitions that can occur during the duration of latency [19]. This method only considers constant delays while adding computational complexity and is heavily dependent on the accuracy of the trained model. Karamzade et al. and Valensi et al. present similar approaches but show that they are feasible for small latencies while diverging with increased latencies due to accumulating prediction errors [20,21]. Wu et al. present a different approach by combining traditional control methods, specifically a backstepping kinematic controller, with receding horizon reinforcement learning to compensate for network-induced delays [22]. Liotet et al. propose an algorithm that learns a belief representation of the current unobserved state, which the agent uses to select actions [23]. However, these examples do not show how their approach performs in variable, nondeterministic latencies, as can be expected in a real-world setup.

2.2. Modelling Latency in 5G Networks

The previous section showed that the performance of RL algorithms can be improved by including latency in the training environment. However, as presented, most of the research does not consider the effects of variable, nondeterministic latency as it is experienced by a real-world device in a wireless network, such as cellular 5G. To include this in the training setup, we must first model the expected latency. This has been tackled by a multitude of previous works. Sawabe et al. developed a probability density function using a sum of multiple Laplace distributions based on real-world measurements of 5G networks [4]. Choi and Kim use machine learning regression models based on one-way delay measurements from a 5G standalone network [24]. Coll-Perales et al. develop models for vehicle-to-everything communication in 5G setups with multiple mobile network operators [25]. Wang et al. and Li et al. provide channel estimation models for more complicated setups, such as millimeter-wave multiple-input multiple-output channels, and setups with reflecting reconfigurable intelligent surfaces [26,27]. This selection of papers demonstrates a wide breadth of research on channel estimation and network modelling. It is therefore feasible to include a realistic modelling of expected latencies into the training environment of RL agents.

2.3. Motivating the Experiments in This Paper

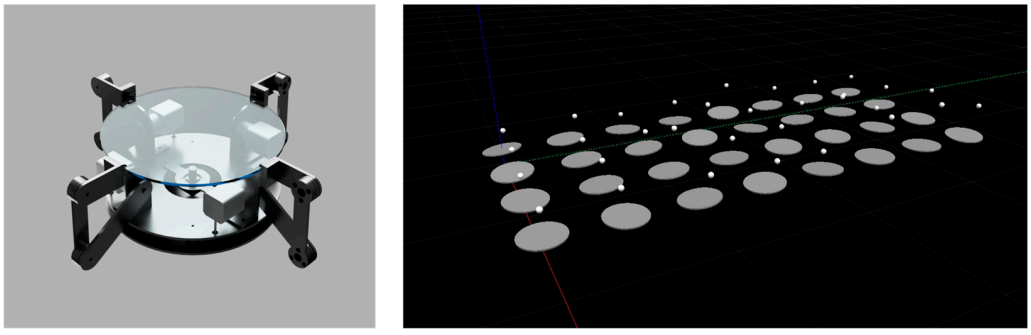
Based on the presented research on previous works for RL training with latencies, the work by Sandha et al. [15] stood out as being comparatively simple yet effective. This paper further investigates an incremental improvement using long short-term memory (LSTM) cells with this approach. This is motivated by previous work from Hausknecht and Stone, who showed that LSTMs perform advantageously in non-Markovian environments [28]. We show that adding the measured latency as inputs to the network with the use of LSTMs is a promising strategy for enabling the agent to handle variable network latencies. To model these latencies, we utilize the probability density function developed by Sawabe et al. [4]. This was chosen as a compromise for its simplicity and real-world applicability while still capturing the nondeterministic nature of the latency in a 5G network.

3. Description of Training Task and Environment

This section will first introduce the RL task that is being considered in this paper. We then introduce the setup of the trained RL agents, followed by a description of the chosen network latency simulation. Finally, the section concludes with a summary of the specific training setups used in this paper.

3.1. The 5G-TSN-Juggler

The 5G-TSN-Juggler (TSN is an abbreviation for time-sensitive networking) is a demonstrator used at the Fraunhofer Institute for Production Technology IPT is used as an exemplary task for the RL control system. The 5G-TSN-Juggler has a glass plate actuated by four arms using stepper motors (refer to Figure 1a). Two stepper motors each are controlled via a wired ethernet-based network and a wireless 5G communication link. The goal of the showcase is to bounce a ping-pong ball on the glass plate in a controlled fashion. Normally, the 5G-TSN-Juggler is used to showcase the deterministic communication capabilities of 5G ultra-reliable low-latency communication using TSN. It was chosen in this paper to contrast different strategies for dealing with communication latency in wireless networking. Kehl et al. present an analysis of TSN over 5G for robotics use cases [29].



(**a**) (**b**)

Figure 1. Depictions of the 5G-TSN-Juggler used as an exemplary RL task in this paper. (**a**) Shows a CAD render of the 5G-TSN-Juggler. Each of the 4 motors can be controlled separately, resulting in 3 degrees of freedom for the glass plate: vertical movement and tilting along the 2 axes defined by opposing motors; (**b**) shows the 32 parallel training instances in Gazebo [23].

The setup of the 5G-TSN-Juggler using the 4 arms controlled by the motors allows the plate to move up and down and tilt along the two axes described by the opposing pairs of arms. Refer to Figure 2 for the different actuations of the 5G-TSN-Juggler.

The following list describes the inputs to the RL agent at each 10 ms. Refer to Figure 2 for a depiction of the corresponding coordinate system.

- Position of the glass plate in the z axis and tilt angles along the x and y axes
- Position vector of the ball in x, y, and z
- Velocity of the glass plate in z and angular velocity of the glass plate in x and y
- Velocity of the ball in x, y, and z
- The RL agent outputs the following values at each time step:
- Force F_z
- Torques T_x and T_y

To train the RL agents in simulation, 32 parallel instances were set up in Gazebo [30] (refer to Figure 1b). Each training episode is initiated using a random position of the ball over the plate and is allowed to continue for either 400 time steps or until the ball drops. The reward function is defined in a way to incentivize the agents to bounce the ping-pong ball at a constant height and centered over the plate. A precise definition of all constraints and the equation for the reward function can be seen in Appendix A.

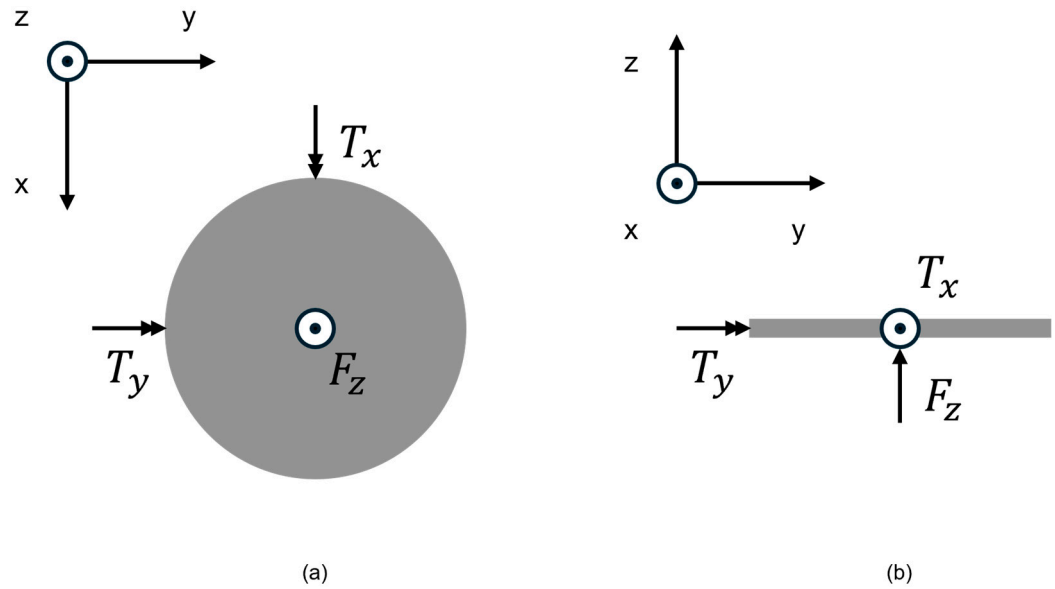


Figure 2. Diagram of the actuation of the simplified 5G-TSN-Juggler glass plate from two perspectives. (**a**) Shows a top-down view; (**b**) shows a side view with the corresponding orientation of the coordinate system. Torques T_x and T_y allow tilting of the glass plate along the x and y axes, which are described by the pairs of opposing arms (see Figure 1a). Force F_z moves the plate along the z axis. Due to the constraints of the system, the plate cannot move in x or y direction.

3.2. Setup of the Reinforcement Learning Agents

The agent of choice for this paper is based on proximal policy optimization (PPO), originally proposed by Schulman et al. [31]. PPO was previously shown to excel in RL tasks where there is a temporal difference between an action taken by the agent and the environment responding with a reward [32]. Although this is normally viewed as the agents' capability for strategic thinking and planning, this is also useful for the situation for communication latency between the agent and environment, as the agent needs to learn to expect latency spikes in the future and therefore leave safety margins. The architecture for the neural networks of both the policy and the value function are depicted in Figure 3. Dedicated hyperparameter optimization is outside the scope of this paper. Further details and additional information on hyperparameters are available in Appendix A.

The agents were trained using temporal difference (TD), TD-lambda (TD(λ)), and Monte Carlo (MC) learning. Note that, although the name TD suggests the exact scenario viewed in this paper, it is unrelated to the difference in time introduced due to communication latency. In the non-Markovian scenario of communication latencies viewed in this

paper, MC is expected to be more robust because it relies on the complete training episodes to update the value function, which allows it to potentially capture the broader context of the environment. However, TD allows for more frequent updates of the value function using the estimates of the value function after every step. This allows TD to converge faster and be more data efficient, using fewer training examples. As an in-between of these two methods, TD(λ) uses a weighted average of several training steps determined using the λ trace decay parameter. Higher values of λ allow for longer-lasting episodes, with the extremes $\lambda = 0$ and $\lambda = 1$ being equivalent to TD and MC learning, respectively [33].

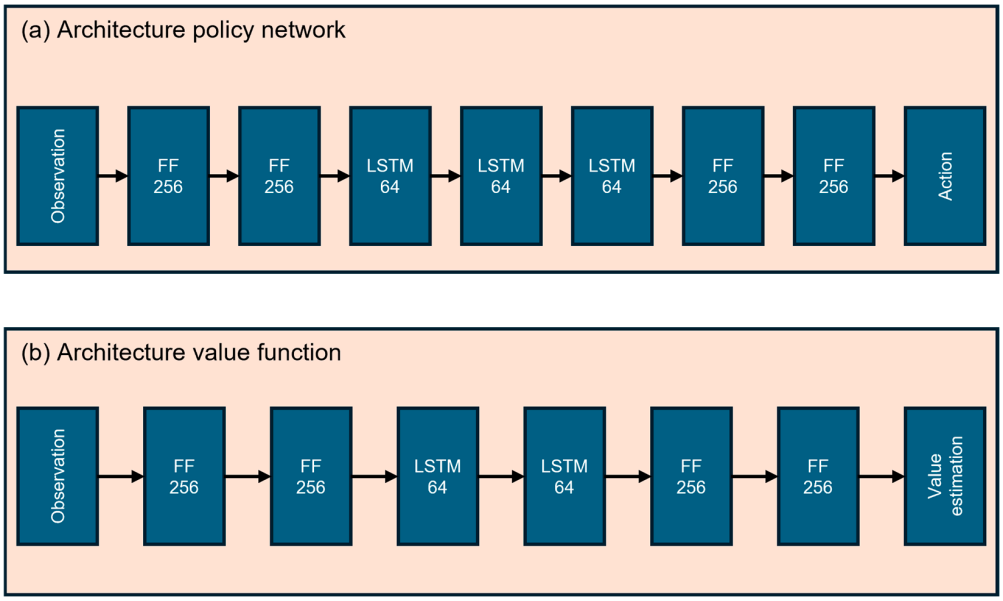


Figure 3. Architecture of the policy (**a**) and value function (**b**) neural network. FF refers to feed forward layers, while LSTM refers to long short-term memory cell-based layers. The number in each block refer to the number of neurons in each layer. All layers used the ReLU activation function.

3.3. Simulation of 5G Network Latency

To simulate the latency expected in a 5G network, we require a probability density function. This paper uses a Laplace mixture model proposed by Sawabe et al., developed specifically for the scenario of mobile, high-frequency communication that is expected in the use case of remote control of mobile robotics using reinforcement learning [4]. Sawabe et al. measured distinct peaks in the latency distribution at 12 ms, 20 ms, 28 ms, 32 ms, 40 ms, and 48 ms. Each of these peaks is modelled using a separate Laplace distribution, which are combined using an additional Laplace distribution to account for outliers. While these latency peaks are comparatively higher than measurements taken in other private 5G industrial campus networks [3], they were empirically measured, and the accuracy of the model was validated [4]. Therefore, this latency distribution can be viewed as a realistic latency distribution for use cases using large-scale private 5G networks or public 5G networks, such as autonomous logistics robots in ports or autonomous driving. Figure 4 gives an overview of the resulting latency distribution.

To include the effects of this latency in the training environment of the agents, the communication mechanism was implemented using a buffer that contained the messages, along with the dispatch time and a latency value sampled from the previously described distribution. At each simulation time step, the newest message with dispatch time plus latency smaller than the simulation timestamp was returned from the buffer. Older messages were deleted as they were no longer relevant. If a new message was generated with a latency that would result in it being sent before an older message already in the buffer, the

older message was also discarded. This means messages that are overtaken are lost, which effectively simulates packet loss in communication networks.

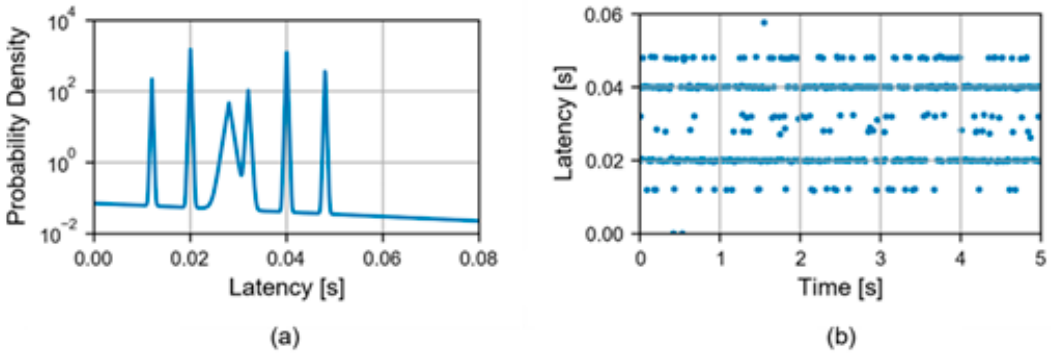


Figure 4. Latency distribution of the 5G network latency simulation used in this paper. (**a**) Shows the probability density function based on the results from Sawabe et al. [3]; (**b**) shows a sequence of samples generated by using the distribution shown in (**a**).

3.4. Description of the Training and Inference Scenarios

As mentioned in Section 3.2, 3 different agents were trained using TD, TD (λ), and MC. For TD (λ) training, the value of λ was set to 0.95. Each of these agents is trained in 3 different scenarios:

1. Baseline without communication latency;
2. With simulated 5G communication latency according to Section 3.3;
3. With simulated 5G communication latency according to Section 3.3 but also receiving the latency of the received message, similar to the setup described by Sandha et al. [15].

The neural networks of the agents in the baseline scenario were initialized with random weights before the beginning of training. The trained baseline agents were then used as the starting point for the training of the scenarios with communication latency. Thus, a total of 9 different agents were trained. The agents were then each evaluated in 4 different inference scenarios:

1. Baseline without communication latency;
2. With simulated 5G communication latency, but with the values halved;
3. With simulated 5G communication latency according to Section 3.3;
4. With simulated 5G communication latency, but with the values doubled.

The different inference scenarios are selected to give us an overview of the agents' performance in the same environment they were trained in, as well as with a previously untrained scenario with less/more latency.

4. Results

This section first describes the evolution of the agents' reward during training, followed by the agents' performance in inference using the different environments, as described in Section 3.4.

4.1. Reward During Training Without Communication Latency

An exemplary evolution of the reward of the three agents during training in an environment without any communication latency is depicted in Figure 5.

The graph suggests two distinct stages of training: one at an average reward of roughly 2000 and the other at an average reward of roughly 4000. Closer examination of the training process shows that the agents reach the first stage by learning to balance the ball on the plate. The reward function is set up in a way where dropping the ball is heavily penalized, so balancing the ball is an effective strategy to avoid a penalty. The second stage is reached

when the agents start to bounce the ball stably. Although there are differences in the average reward value at the end of training, all agents learned to bounce the ball during this baseline comparison. A closer inspection of the differences is outside of the scope of this paper.

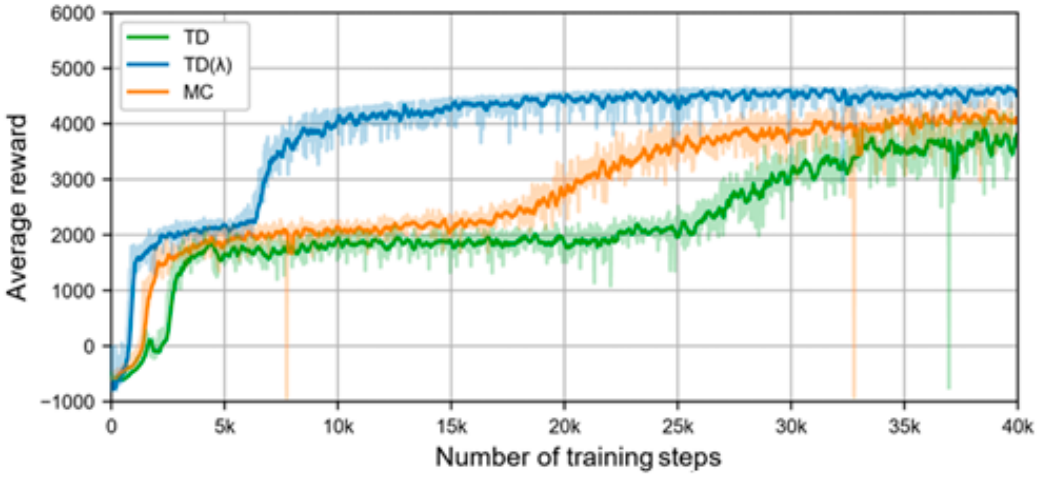


Figure 5. Exemplary graph of the reward during training of the 3 agents in an environment without communication latency. The two distinct stages of training at roughly 2000 and 4000 reward are the agents learning to balance the ball on the plate first and then learning to bounce it stably.

4.2. Reward During Training with Simulated 5G Communication Latency

An exemplary evolution of the reward of the three agents during training with simulated 5G communication latency is depicted in Figure 6.

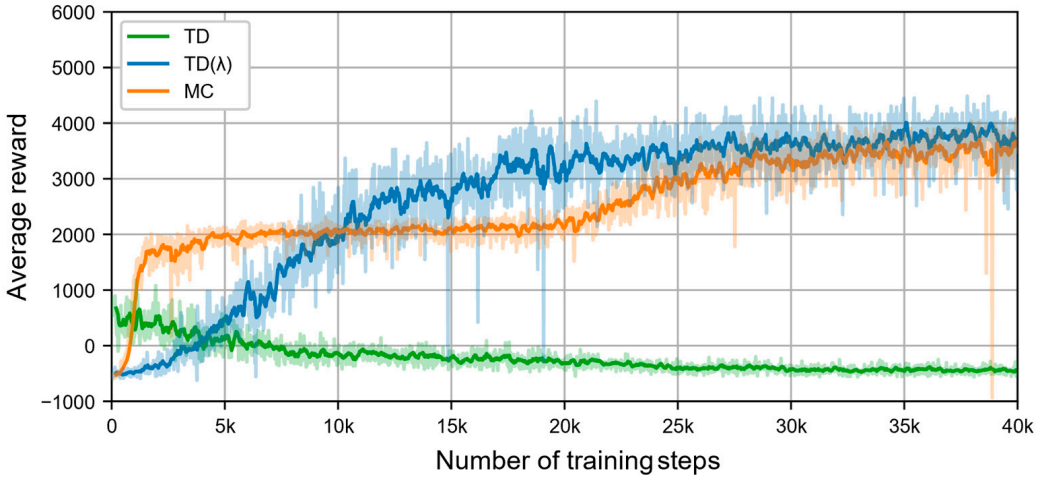


Figure 6. Exemplary graph of the reward during training of the 3 agents in an environment with simulated 5G communication latency. One can clearly see that the agent trained with pure TD training failed to learn to even balance the ball.

The graph clearly shows that the agent trained with TD was not able to learn to even balance the ball (compared to Section 4.1, the agent fails to reach initial reward plateau of 2000). The other two agents were able to bounce the ball. Furthermore, the general trajectory of the MC agent is similar to the baseline scenario but with a lower final reward, as expected. The TD (λ) agent, in contrast, does not seem to reach a local minimum at balancing the ball in this scenario and outperforms the MC agent in the end.

4.3. Reward During Training with Simulated 5G Communication Latency with Latency as Input

An exemplary evolution of the reward of the three agents with additional latency input during training with simulated 5G communication latency is depicted in Figure 7.

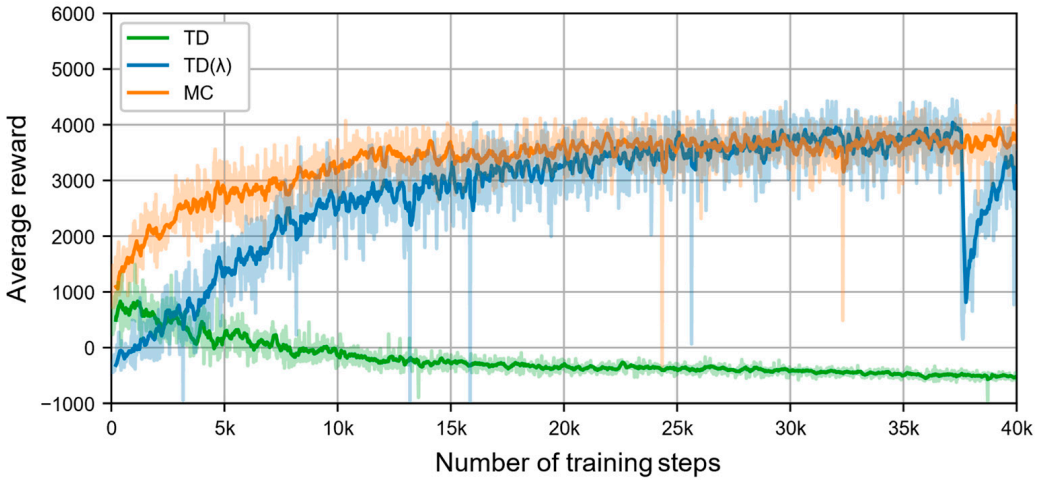


Figure 7. Exemplary graph of the reward during training of the 3 agents in an environment with simulated 5G communication latency and the agents receiving the measured latency as an additional input. One can clearly see that the agent trained with TD failed to learn to even balance the ball. In contrast to previous scenarios, there is some instability in the agent trained with $\lambda = 0.95$.

Similar to the previous scenario, the TD agent fails to learn to even balance the ball (compared to Section 4.1, the agent fails to reach the initial reward plateau of 2000). In contrast to the baseline scenario, both agents now continuously improve to bouncing the ball instead of plateauing at a local minimum by balancing the ball for an extended period. Furthermore, some of the experiments performed with $\lambda = 0.95$ showed some instability via drops in reward during training. This is in line with the expectation set up by the theory of increased stability with MC training (refer to Section 3.2).

4.4. Results of Inference in Different Scenarios

Figure 8 depicts the results of the different inference scenarios described in Section 3.4. The table is structured as follows:

- Each row represents a different agent as trained using the different methods described in Section 3.4. They are grouped by the labels to the left, based on whether they were trained without latency, with latency, and with latency and receiving it as an additional input.
- The columns represent the different inference scenarios, as described in Section 3.4. The columns are grouped by the labels at the top. These two groups represent the results from the same inference scenarios, with the left group showing the average reward value and the right group showing the ratio of episodes where the ball was dropped.
- The columns where the training scenario matched the inference scenario, i.e., agents trained without latency are running inference without latency, and agents trained with latency are running inference with the same amount of latency as they were trained in, are highlighted with a blue border around the cells.
- In the left column group, darker green hues represent a higher average reward value and, therefore, better inference performance.
- In the right column group, darker red hues represent a higher ratio of inference episodes where the ball is dropped and, therefore, worse inference performance.

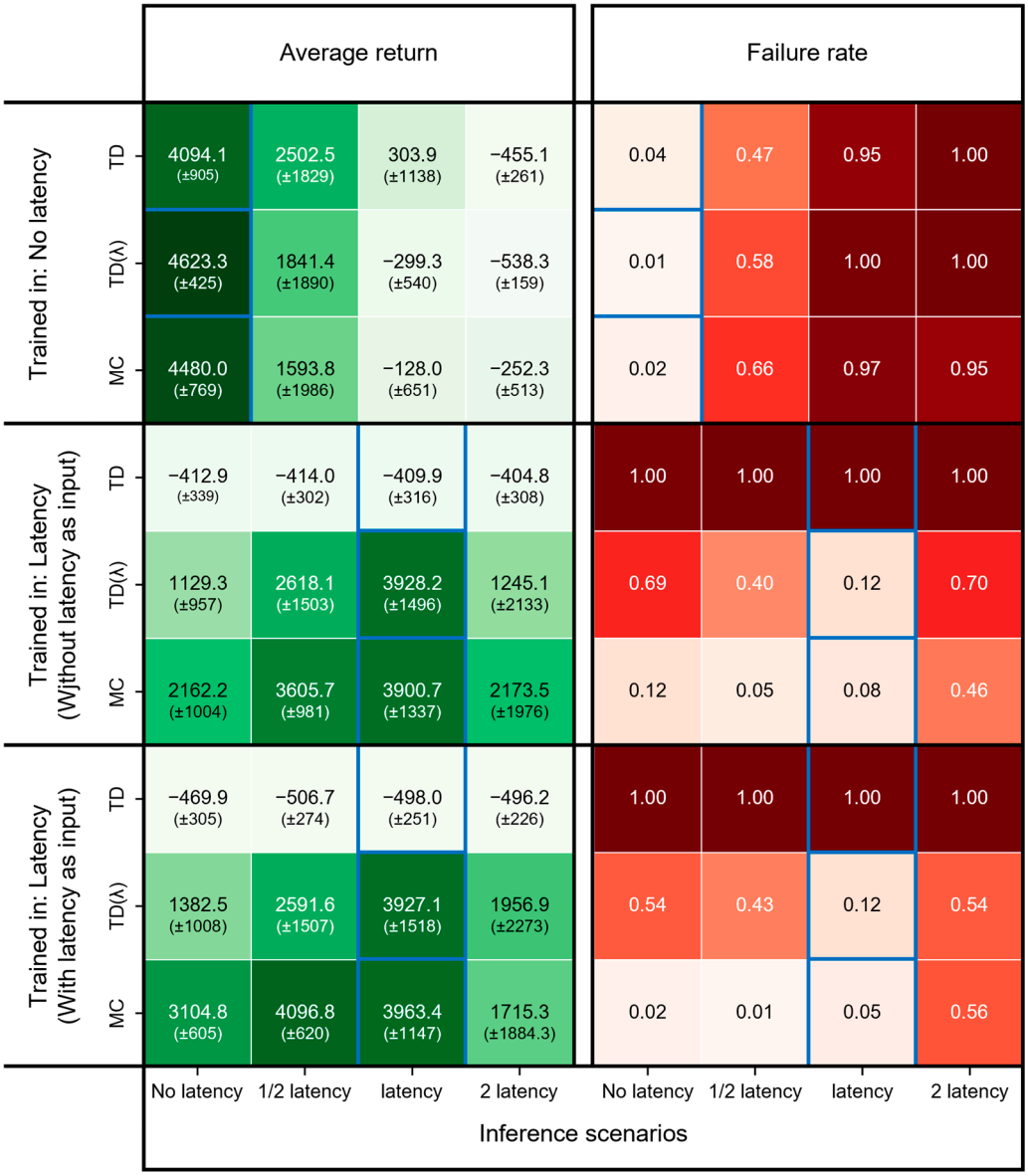


Figure 8. Inference results of the best-performing agents during training in the different inference scenarios. Each combination of agent and inference scenario was run for 512 episodes, with the figure showing the average values. Failure rate refers to the ratio of episodes were the agent dropped the ball. "1/2 latency" and "2 latency" refer to the scenario were the sampled latency from the distribution described in Section 3.3 was halved and doubled, respectively. The blue squares denote the scenarios where the inference matched the training setup.

The table shows a few general patterns, as follows:

- The TD agent failed to learn to bounce the ball in any scenario with communication latency. These results indicate that pure TD training is insufficient for scenarios with communication latency.
- As expected, the introduction of communication latency leads to a degradation in agent performance (compare columns 2, 3, and 4 to column 1 in both column groups in the first three rows). This can be seen in both the average reward and the failure rate.
- Training the TD (λ) and MC agents with latency drastically improved their performance in inference scenarios with communication latency; see column 3 in both column groups for rows 4–9.
- The TD (λ) and MC agents trained with latency performed significantly better during inference with lower latency values (compare column 2 in both groups).

- The inference scenarios with doubled latency show generally decreased performance; see column 4 in both groups. However, the TD (λ) and MC agents trained with latency performed significantly better than the ones trained without input latency.

Additionally, the table shows that the TD (λ) agents perform best in the inference scenario with the same latency as they were trained in. Notably, the MC agent trained with input latency can perform better in scenarios with lower input latency. The MC agent trained with latency, which also receives the measured latency as an additional input, reaches the same failure rate as in the baseline scenario, even though the reward is lower.

5. Discussion

This section will discuss and interpret the results shown in Section 4, establishing some general patterns emerging from both the training and the inference scenarios.

One immediate interpretation to draw is that pure TD training fails to learn to bounce the ball if there is communication latency present. This can be explained using the fact that the reward function in TD learning is updated at each training step and therefore fails to capture the context of latency, which is inherently only visible by viewing a multitude of consecutive time steps.

Both TD (λ) and MC agents successfully learned to bounce the ball in all scenarios. Based on this, we conclude that the RL setup of PPO with LSTM-based neural networks combined with TD (λ) or MC learning methods is effective in handling variable communication latencies in networked control systems. The success of the TD (λ) and MC agents can be attributed to their ability to consider longer temporal dependencies during training. TD (λ) introduces an eligibility trace that allows the agent to update its value function based on a combination of immediate and future rewards, weighted by the trace decay parameter λ. This enables the agent to learn from sequences of actions and observations, effectively capturing the effects of communication latency. Similarly, MC methods rely on the total return from complete episodes, inherently incorporating the cumulative impact of delays into the learning process. In contrast, the standard TD method updates the value function at each time step using only the immediate reward and the estimated value of the next state. This makes it less capable of handling the non-Markovian properties introduced by communication latency, as it does not account for the delayed effects of actions on future states.

Our experiments also demonstrated that providing the measured latency as an additional input to the agent improves performance in environments with variable communication delays. This aligns with the findings of Sandha et al. [15], highlighting the importance of latency awareness in training RL agents for networked control systems. By incorporating the measured latency, the agent can adjust its actions to compensate for delays, leading to more robust control policies.

The use of LSTM cells in the neural network architecture further enhanced the agents' ability to manage communication latency. LSTMs are designed to capture long-term dependencies in sequential data, making them well suited for environments where the Markov property is violated due to delays. By maintaining a memory of previous inputs and states, the LSTM-based agents can infer and anticipate the impact of communication latency on the control process.

Another interesting observation that can be taken from the training data is that pretraining the agents on a baseline scenario without communication latency did not significantly impact the starting performance of the agents or the convergence speed of learning. While this is somewhat to be expected due to the completely different dynamics of the environment and system, this is still notable and can allow for ignoring pretraining in the future.

Analyzing the inference scenarios, we observed that agents trained with latency awareness performed better when the latency conditions during inference matched those experienced during training. However, their performance degraded when faced with latency conditions that were significantly different (either halved or doubled) from the training environment. This suggests that while the agents developed strategies to cope with the specific latency distribution they were trained on, their ability to generalize to drastically different latency scenarios is limited.

This limitation underscores the importance of incorporating a diverse range of latency conditions during training to improve the agents' robustness. Techniques such as domain randomization, where latency is varied over a wide range during training, could help agents develop more adaptable policies that perform well under varying network conditions. Another key observation is that the agents' performance, measured both by average reward and failure rate, is closely tied to the accuracy of the latency simulation. Agents trained with a latency model that accurately reflects real-world conditions are more likely to perform effectively in practical applications. Therefore, investing effort in developing realistic latency models is crucial for the successful deployment of RL-based control strategies in networked systems.

6. Conclusions and Outlook

In this paper, we investigated the potential of reinforcement learning (RL) to develop adaptive control strategies for networked control systems operating under variable communication latency, specifically within industrial 5G networks. Using the 5G-TSN-Juggler as a demonstrator, we trained RL agents to control a system that requires precise timing despite the challenges introduced by network-induced delays.

Our approach involved training RL agents using proximal policy optimization (PPO) with long short-term memory (LSTM) networks to handle the variable latency inherent in 5G networks. By comparing different learning methods—temporal difference (TD), TD (λ), and Monte Carlo (MC)—we evaluated how well each method adapted to the communication delays typical of 5G networks. The agents were trained and tested under various simulated latency conditions that reflect realistic 5G network performance, including latency spikes and variability due to factors such as device mobility and network traffic.

The results demonstrated that agents trained with latency awareness—particularly those using TD (λ) and MC methods and receiving measured latency as an additional input—successfully managed to control the system despite the variable communication delays inherent in 5G networks. The inclusion of LSTM networks allowed the agents to capture temporal dependencies and adapt their control strategies in response to latency fluctuations. Conversely, agents trained without considering latency or using standard TD methods were unable to cope with the delays, underscoring the necessity of incorporating latency information and appropriate learning strategies when designing control systems over 5G networks.

Our findings emphasize the potential of combining advanced RL techniques with 5G communication capabilities to enable robust and adaptive control in industrial applications. This has significant implications for the deployment of 5G in scenarios such as teleoperation of machinery, flexible robotic manufacturing cells, and autonomous vehicles, where network reliability and latency are critical factors.

Further research is required to conclude that the patterns shown in this paper can be generalized to other tasks and latency distributions. As the literature review in Section 2 shows, there are multiple different approaches to dealing with communication latency in the research field of RL. Although our results broadly correlate with the results of Sandha

et al. [15], there needs to be a meta-analysis of the different approaches using the same task and different latency distributions. Furthermore, we have not performed any dedicated hyperparameter tuning and computational complexity analysis in this paper. This would give additional insight into how far our approach can improve performance using optimal hyperparameters, as well as an efficiency comparison with other approaches.

Additional research is required to determine the maximum latency an agent can tolerate while successfully performing a task within predefined tolerances during training. Since 5G networks allow for advanced quality of service settings for the network, this would allow a network operator to assign more/less network resources to high- and low-priority tasks, respectively, ensuring that a high-priority task receives a preferred network access for lower and more reliable latency.

Another avenue for future work is the real-world deployment and testing of these RL agents in live 5G network environments. Although the simulated communication latencies used in this paper are motivated by real-world data, they fail to capture additional challenges introduced by the cellular characteristics of 5G, such as rapid latency changes due to handovers or network congestion.

Author Contributions: Conceptualization, A.G.; Methodology, A.G. and N.B.; Software, N.B.; Validation, A.G.; Formal analysis, A.G. and N.B.; Writing—original draft, A.G.; Writing—review & editing, P.E.K.; Supervision, A.G. and P.E.K.; Project administration, R.H.S. All authors have read and agreed to the published version of the manuscript.

Funding: Co-funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the other granting authorities. Neither the European Union nor the granting authority can be held responsible for them. The TARGET-X project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101096614.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

Appendix A

This appendix contains additional details for the simulation setup and the reward function used in this paper to enable a reproduction of our experiments and results.

Appendix A.1. Supplemental Information About the Simulation of the 5G-TSN-Juggler

Table A1 provides some additional information about the simulation setup that is unrelated to results and the conclusions drawn in this paper.

Table A1. Supplemental information about the simulation and training setup.

Description	Value
Acceleration due to gravity	9.81 m/s^2
Mass of the ping-pong ball	27 g
Diameter of the ping-pong ball	0.04 m
Diameter of the glass plate	0.3 m
Limits of the glass plate's vertical movement	0 m below and 0.07 cm above the resting position
Limits of the glass plate's tilting	−0.085 rad to 0.085 rad from parallel to ground

Table A1. *Cont.*

Description	Value
Initial offset of the ball from the center of the glass plate at the beginning of training	Uniformly sampled for both axis [−0.05 m, 0.05 m]
Initial vertical offset the ball from the glass plate at the beginning of training	Uniformly sampled [0.2 m, 0.5 m]
The initial velocity of the ball	Uniformly sampled for both axis [−0.1 m/s, 0.1 m/s]
Restitution of the ball	0.9
Restitution threshold	0
Friction coefficient of the ball	0.5

Appendix A.2. Detailed Definition of the Reward Function

The agent is considered to have dropped the ball if its center is at a horizontal position (x and y positions) outside the cylinder defined by the circle of the glass plate in its resting position. The ball is also considered to be dropped if its center is below (z position) 0.04 m. A penalty of 1000 points is applied for dropping the ball.

The main objective is to keep the ball bouncing. This is computed using the following equation:

$$g \cdot z_b + \frac{1}{2} \cdot \dot{z}_b^2 - g \cdot h_d \tag{A1}$$

The idea behind this equation is, neglecting losses due to drag, the sum of the potential energy $g \cdot z_b$ and kinetic energy $1/2 \cdot \dot{z}_b^2$ of the ball should remain equal to the potential energy at the peak of the bounce at a desired height h_d. Any deviation from that means that the agent bounced the ball too strongly or too weakly.

The secondary objective is to keep the ball centered above the glass plate. Further objectives are added to penalize erratic movements by the agent.

The reward value of each objective is computed using the radial basis function (RBF). The width of the RBF is determined by the parameter ζ_i. The weight of each objective's reward is defined by the parameter w_i.

Table A2. Parameters of the weighted sum of the reward function.

Description	ζ_i.	w_i
Difference of target energy	0.5	8
Ball x position	1000	0.5
Ball y position	1000	0.5
Torque of each axis	0.1	0.25
Vertical force	0.01	0.25
Vertical offset plate of resting position	1000	3
Tilt of plate in each axis	100	1
Radial velocity plate in each axis	10	0.125

Appendix A.3. Additional Information on Network Architecture and Hyperparameters

The following list contains additional information and hyperparameters used during the training of the agents described in this paper:

- All kernels in the neural networks were initialized using the Tensorflow variance scaling initializer with scale 2.0 and otherwise default arguments [34].
- All layers used the ReLU activation function.

- Training used the Adam optimizer as described by Kingma et al. [35]. The setup used the default Tensorflow values for learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$ [36].
- The PPO agent was implemented using the Tensorflow PPOClipAgent implementation [37]. The default hyperparameter values were used during training.

References

1. What is Industrie 4.0? Available online: https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html (accessed on 4 February 2025).
2. Zhang, X.-M.; Han, Q.-L.; Ge, X.; Ding, D.; Ding, L.; Yue, D.; Peng, C. Networked control systems: A survey of trends and techniques. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1–17. [CrossRef]
3. Mehmet. 5G-Industry Campus Europe-5G-ACIA. Available online: https://5g-acia.org/testbeds/5g-industry-campus-europe/ (accessed on 4 February 2025).
4. Sawabe, A.; Shinohara, Y.; Iwai, T. Delay Jitter Modeling for Low-Latency Wireless Communications in Mobility Scenarios. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 2638–2643, ISBN 978-1-6654-3540-6.
5. Lunze, P.J. *Regelungstechnik 1*; Springer: Berlin/Heidelberg, Germany, 2020; ISBN 978-3-662-60745-9.
6. Yasunobu, S.; Sasaki, R. An auto-driving system by interactive driving knowledge acquisition. In Proceedings of the SICE 2003 Annual Conference, Fukui, Japan, 4–6 August 2003; Society of Instrument and Control Engineers: Tokyo, Japan, 2003. ISBN 0-7803-8352-4.
7. Cristea, S.; Prada, C.D.; Keyser, R.D. Predictive control of a process with variable dead-time. *IFAC Proc. Vol.* **2005**, *38*, 309–314. [CrossRef]
8. Wu, Q.; Zhan, S.S.; Wang, Y.; Wang, Y.; Lin, C.-W.; Lv, C.; Zhu, Q.; Huang, C. Variational Delayed Policy Optimization. In Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 10–5 December 2024.
9. Nsnam. ns-3. Available online: https://www.nsnam.org/ (accessed on 9 February 2025).
10. Singh, B.; Kumar, R.; Singh, V.P. Reinforcement learning in robotic applications: A comprehensive survey. *Artif. Intell. Rev.* **2022**, *55*, 945–990. [CrossRef]
11. Rupam Mahmood, A.; Korenkevych, D.; Komer, B.J.; Bergstra, J. Setting up a Reinforcement Learning Task with a Real-World Robot. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 4635–4640.
12. Xie, Z.; Berseth, G.; Clary, P.; Hurst, J.; van de Panne, M. Feedback Control for Cassie With Deep Reinforcement Learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1241–1246.
13. Travnik, J.B.; Mathewson, K.W.; Sutton, R.S.; Pilarski, P.M. Reactive Reinforcement Learning in Asynchronous Environments. *Front. Robot. AI* **2018**, *5*, 79. [CrossRef] [PubMed]
14. Andrychowicz, O.M.; Baker, B.; Chociej, M.; Józefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **2020**, *39*, 3–20. [CrossRef]
15. Sandha, S.S.; Garcia, L.; Balaji, B.; Anwar, F.; Srivastava, M. Sim2Real Transfer for Deep Reinforcement Learning with Stochastic State Transition Delays. *Conf. Robot Learn.* **2021**, *155*, 1066–1083.
16. Chen, B.; Xu, M.; Li, L.; Zhao, D. Delay-aware model-based reinforcement learning for continuous control. *Neurocomputing* **2021**, *450*, 119–128. [CrossRef]
17. Bouteiller, Y.; Ramstedt, S.; Beltrame, G.; Pal, C.; Binas, J. Reinforcement Learning with Random Delays. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
18. Derman, E.; Dalal, G.; Mannor, S. Acting in Delayed Environments with Non-Stationary Markov Policies. 2021. Available online: http://arxiv.org/pdf/2101.11992v4 (accessed on 11 February 2025).
19. Yao, Z.; Florescu, I.; Lee, C. Control in Stochastic Environment with Delays: A Model-based Reinforcement Learning Approach. In Proceedings of the International Conference on Automated Planning and Scheduling, Banaff, AB, Canada, 1–6 June 2024.
20. Karamzade, A.; Kim, K.; Kalsi, M.; Fox, R. Reinforcement Learning from Delayed Observations via World Models. *Reinf. Learn. J.* **2024**, *5*, 2123–2139.
21. Valensi, D.; Derman, E.; Mannor, S.; Dalal, G. Tree Search-Based Policy Optimization under Stochastic Execution Delay. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
22. Wu, R.; Xu, X.; He, M.; Liu, D.; Zhang, X. Learning predictive control of wheeled mobile robot with communication delays. *J. Phys. Conf. Ser.* **2024**, *2797*, 12026. [CrossRef]

23. Liotet, P.; Venneri, E.; Restelli, M. Learning a Belief Representation for Delayed Reinforcement Learning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8, ISBN 978-1-6654-3900-8.
24. Choi, S.; Kim, C. A Study on Latency Prediction in 5G network. In Proceedings of the 2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN), Paris, France, 4–7 July 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 713–715, ISBN 979-8-3503-3538-5.
25. Coll-Perales, B.; Lucas-Estañ, M.C.; Shimizu, T.; Gozalvez, J.; Higuchi, T.; Avedisov, S.; Altintas, O.; Sepulcre, M. End-to-End V2X Latency Modeling and Analysis in 5G Networks. *IEEE Trans. Veh. Technol.* **2023**, *72*, 5094–5109. [CrossRef]
26. Li, X.; Zhao, J.; Chen, G.; Hao, W.; da Costa, D.B.; Nallanathan, A.; Shin, H.; Yuen, C. STAR-RIS Assisted Covert Wireless Communications with Randomly Distributed Blockages. *IEEE Trans. Wirel. Commun.* **2024**. [CrossRef]
27. Wang, H.; Xiao, P.; Li, X. Channel Parameter Estimation of mmWave MIMO System in Urban Traffic Scene: A Training Channel-Based Method. *IEEE Trans. Intell. Transport. Syst.* **2024**, *25*, 754–762. [CrossRef]
28. Hausknecht, M.J.; Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. In Proceedings of the AAAI Fall Symposia, Arlington, VA, USA, 12–14 November 2015.
29. Kehl, P.; Ansari, J.; Jafari, M.H.; Becker, P.; Sachs, J.; König, N.; Göppert, A.; Schmitt, R.H. Prototype of 5G Integrated with TSN for Edge-Controlled Mobile Robotics. *Electronics* **2022**, *11*, 1666. [CrossRef]
30. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 2149–2154, ISBN 0-7803-8463-6.
31. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
32. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
33. Sutton, R.S.; Barto, A. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA; London, UK, 2020; ISBN 9780262039246.
34. TensorFlow. tf.compat.v1.variance_scaling_initializer | TensorFlow v2.16.1. Available online: https://www.tensorflow.org/api_docs/python/tf/compat/v1/variance_scaling_initializer (accessed on 15 March 2025).
35. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
36. TensorFlow. tf.keras.optimizers.Adam | TensorFlow v2.16.1. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (accessed on 15 March 2025).
37. TensorFlow. tf_agents.agents.PPOClipAgent | TensorFlow Agents. Available online: https://www.tensorflow.org/agents/api_docs/python/tf_agents/agents/PPOClipAgent (accessed on 15 March 2025).