Analysis of Network Performance over Deep Reinforcement Learning Control Loops for Industry 4.0

Guilherme T. T. Bernardo[1], **Gilson Miranda Jr.**[12], **Daniel F. Macedo**[1]

[1]Computer Science Department - Universidade Federal de Minas Gerais (UFMG)
Zip Code 31.270-901 – Belo Horizonte – MG – Brazil

[2]University of Antwerp - imec, IDLab, Faculty of Applied Engineering
Zip Code 2000 – Antwerp – Belgium

{guilhermebernardo,gilsonmiranda,damacedo}@dcc.ufmg.br

Abstract. *One of the three core use cases of 5G is ultra-reliable low-latency communications, which encompasses remote operation for Industry 4.0. It is expected that autonomous robots, driven by artificial intelligence agents running on the cloud, will operate without human assistance. This paper investigates how the network performance affects the behavior of AI-based remote operation control that is based on deep reinforcement learning. We investigate two separate aspects:* (i) *the performance of a pre-trained model on ideal network conditions and* (ii) *how the varying network conditions impact the agent performance in the environment. The results show the influence of different network quality conditions on agent's performance, as well as the benefit of using agents pre-trained on ideal conditions.*

Resumo. *Dentre os três principais casos de uso das redes 5G está a comunicação ultra-confiável de baixa latência, que inclui a operação remota para a Indústria 4.0. Nesse tipo de aplicação, robôs autônomos são capazes de operar sem intervenção humana, guiados por sistemas de inteligência artificial executados em nuvem. Este artigo investiga dois diferentes aspectos:* (i) *a performance de um modelo pré-treinado em condições de rede ideais; e* (ii) *como variações de desempenho da rede influenciam o desempenho dos modelos no ambiente. Os resultados mostram a influência de diferentes condições de qualidade da rede no desempenho do agente, assim como o benefício de utilizar agentes pré-treinados em condições ideais.*

1. Introduction

As humans and robots work ever closer in an industrial context, automation tools become a differential in the market by making production processes more efficient [Buchner et al. 2012]. Workers and machines are also becoming more tightly integrated with the Internet of Things (IoT), making the control of machines an increasingly complex task [Huang et al. 2020]. Industry 4.0 is characterized as a movement that allows factories to be intelligent enough to anticipate and identify problems and changes in production [Faheem et al. 2018]. To keep this momentum and allow more devices to be interconnected, the network infrastructure also evolve. Advances in information systems allow operators in the factory to have increasingly up-to-date and interactive information

[Papcun et al. 2018]. In the context of Industry 4.0, with more devices to coordinate, operators must create dynamic workflows that can be remotely managed. However, current 4th Generation (4G) mobile networks and other communication technologies cannot meet the Industry 4.0 demands for high data rate, high reliability, high coverage, and low latency [Cheng et al. 2018]. This makes it difficult to develop and maintain robust systems.

The application of intelligent control using Machine Learning (ML) methods has been investigated for many years, aiming at enabling the autonomous control of complex systems that operate in dynamic environments [Harris 1994, Lobbrecht and Solomatine 2002]. Currently, studies in 5G Ultra Reliable Low Latency Communication (URLLC) are paving the way to the remote control of moving machinery (e.g., robots, rovers, vessels) and lowering costs by offloading complex tasks to more capable nodes in the cloud [Civerchia et al. 2020, Liu et al. 2017]. This type of remote control allow the deployment of robots into hazardous environments while humans can operate them from safe installations [Yang et al. 2008]. With the combination of ML-based control for autonomous agents and the reliable communication provided by 5G, the deployment of autonomous control agents that interact remotely with the controlled entity can be envisioned in the near future.

With the current context of interconnected applications, severe limitations can occur in an environment with hundreds of devices managed by a local control architecture, e.g., challenges related to controlling multiple devices and keeping all updated [Nakimuli et al. 2021]. One approach to overcome this problem is to migrate the intelligence to external servers outside the device in a distributed architecture. In this way, it is possible to dedicate exclusive servers for data storage and processing without overloading the system. The remote control architecture requires strict performance constraints on the links between controller and actuators. System performance may be adversely affected if the network cannot meet low latency and low packet loss ratio. Thus, the quality of the network can affect the Artificial Intelligence (AI) decision when running remotely, e.g., when a packet is lost in data traffic and the agent takes an unexpected action.

The main contribution of this work concerns the evaluation of the effects of network metrics on the quality of actuation of remote control loops. More specifically, we investigate how network performance impacts a control loop based on deep Reinforcement Learning (RL). We use a robotic arm as use case, and the rewards and task completion time from the RL agent as metrics to evaluate two different situations. First, when learning occurs over the network. Second, when the agent is pre-trained offline, then deployed to act over a network. The offline pre-trained agent offers a baseline to analyze the effect of the different network impairments on the learning of the agent trained directly over the network. The agent trained offline faces a different condition when deployed in the network, as the action taken is executed remotely under sub-optimal network conditions. Nevertheless, the agent trained offline was still able to complete the tasks faster and with higher rewards.

This paper is organized as follows: Section 2 details the related work. Section 3 describes the simulation infrastructure and the testing parameters, while Section 4 presents the results obtained with the agents using offline and online training. Finally, Section 5 presents our conclusions and future work.

2. Related works

IoT technology is being extensively used in industries [Atzori et al. 2010]. Industrial applications require very high transmission speeds and are extremely sensitive to network delays. In a closed control loop, interlocking, monitoring, and supervision applications, there are tolerable delays in the range of milliseconds [Akpakwu et al. 2017, ETSI 2020]. However, it is possible to expand the analysis of the application of IoT to other areas besides industry. Those applications, known in 5G as URLLC, have latency and jitter as the two main Quality of Service (QoS) metrics to be observed [Nasrallah et al. 2018].

As delay and jitter are the most important QoS metrics for real-time services [Kunst et al. 2019], they deserve special attention when evaluating the performance of networked control systems. In the work by Kunst et al. [2019] , the authors maintained jitter values between 25 ms and 47 ms, leading to the conclusion that the application used was able to guarantee QoS considering the delay and jitter metrics. Chen et al. [2004] proposed numerical measures which can quantitatively represent the QoS requirements of various applications with different service characteristics, concluding that there cannot be critical data error and loss when controlling critical industrial processes.

In order to understand the challenges in terms of cognition complexity over mobile networks, Luo et al. [2021] proposed an AI-powered mobile network architecture and discussed the use of AI in this architecture to make smarter decisions in different application domains. Through a deep learning based model that integrates cognition with decision, they could provide QoS by preventing network congestion using data from a China telecom operator. They showed that AI has offered tremendous opportunities for network operators to operate over the network, with the view to better satisfy user requirements and significantly reduce capital and operational expenditures.

Jiang et al. [2020] developed an AI-assisted framework for wireless networks, considering multi-agents operation over a RL point of view. They cloud analyse the latency optimization challenge and establish metrics (e.g, packet transmission reliability) to test high-level autonomous driving scenarios. They demonstrated that information latency is more critical in 5G vertical applications than in conventional URLLC.

Rodriguez et al. [2021] presented an experimental framework which consist on the use and performance optimization of 5G-integrated autonomous mobile robots. They used prototyping tools consisting of a network traffic sniffer, a 5G emulator, and a wireless multi-access gateway. When they applied this framework, they focused the overall 5G control and edge-cloud operation of the robots. They observed that the 5G operation of the robots was superior in terms of control-loop reliability to the one of Wi-Fi 6 and the potential of 5G for reliably operating autonomous mobile robots.

Marques et al. [2018] analysed and presented the impact of software-defined networks and cloud computing technologies in systems running real-time applications with low end-to-end latency and high bandwidth requirements. Thus, to achieve URLLC in wireless networks to support robot mobility, they proposed and implemented a splitting of the WiFi architecture functionalities that provides multiconnectivity and redundant communication by offering seamless mobility and failover resilience. They showed that the robot does not lose communication with the cloud, and the throughput degradation is very

small if compared to the handover processes in classic Wi-Fi network. Hence, the results show that the architecture presents a good performance in the recovery to failures, being able of guaranteeing communication at any time.

Similarly, Liberato et al. [2018] proposed a networking architecture to facilitate and enhance network programmability, which is an important enabler for 5G networks. They argued that compared to the 4G network technology, 5G is expected to provide more throughput even with more connected devices, whereas latency should be reduced 30-50 times, especially for URLLC clients. As results, their prototype demonstrated that residue defined networking architecture is able to offer carrier grade protection achieving ultra-reliable communication with sub-milliseconds for failure recovery, which is an important use case in Industry 4.0.

The use of a remote AI system to control Automated Guided Vehicles (AGV) is another application in which timely communication is crucial. Nakimuli et al. [2021] performed several experiments to investigate all feasible scenarios to deploy a remote-controlled AGV using 5G. They provided evidence of the improved performance of 5G over 4G through in a platform which run and validate industrial use cases. Furthermore, when compared to the 4G radio condition, the use case using 5G presented a better control of the AGV in its trajectory, having better tolerance in terms of packet losses and delays.

Beltrão and Pires [2019] propose a case study in a coal recovery boiler of a cellulose factory, in which the application of drones combined with AI and cloud computing for visual inspection is analyzed. Hence, the drone was able to perform autonomous visual inspection in the pulp industry autonomously, delivering very high resolution images and videos in an easy-to-interpret manner and in greater detail. They concluded that this technology drastically reduces the exposure of workers to accident risks, makes field operation faster and generate improvements in asset management processes.

In this study we investigate the effects of network performance on a URLLC application from the point of view of the AI algorithms. Previous works attempted to provide maximum and minimum values for network metrics that guarantee the operation of the applications. This work, on the other hand, quantifies the amount of degradation on the intelligent control loop for different network metrics. We quantitatively show the performance comparison of online and offline trained agents, both from the ML point of view, when analyzing the agent-environment loop rewards obtained, and from the end-user perspective of the application, when analyzing the mean time for the agent to achieve its goal.

3. Test environment description

In this work, we developed a simulator to evaluate the effects that QoS plays in the remote control of AI-based control-loops. The developed simulator considers both the application side and the network side. The description of the simulator, as well as the experimental methodology is shown below. The simulator is an open source software[1].

3.1. Simulated network topology

The topology used for the simulations is composed by two Virtual Machines (VMs), one containing the client, while the other contains the environment, connected

[1] `https://github.com/guilhermebernardo/AI-Based-Control-Loops-Sim`

by a link with impairments emulated using the Traffic Control (TC) Network Emulator (NetEm)[2] tool for Linux. The trained agent from Stable Baselines 3[3] is deployed at the *Controller* VM, and the actions are executed at the *Actuator* VM running an OpenAI Gym[4] environment. The agent's actions are transmitted through network using UDP, so we can clearly observe the effects of the network impairments when error recovery methods from TCP are not employed. The better the network for transporting information, the fewer errors should be observed in the agent's performance. Figure 1 gives an overview of the architecture. The agent decides an action and transmits to the actuator, which in turn, execute the action in the environment and reply with the new state, the reward associated with the action, informs whether the episode (or the task) is done or not, and transmits further control information.

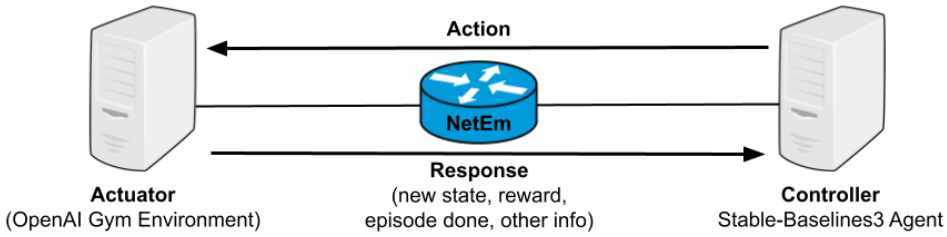

Figure 1. Simulator network architecture model

To evaluate agent performance in the first scenario, we considered a latency from 1 ms to 50 ms, since industrial control applications have delay bounds up to a millisecond [ETSI 2020]. Kunst et al. [2019] maintained jitter values between 25 ms and 47 ms, so we used it in the simulator a threshold value to guarantee QoS in 50 ms. In this work, we consider a threshold rate $\leq 1\%$ for packet loss and corruption, as Chen et al. [2004] quantitatively represented the QoS requirements of various applications. TC-NetEm was used to simulate delays, jitter, packet corruption, packet loss, and packet reordering.

3.2. Software components

To emulate an industrial application, we used the OpenAI Gym environments, which is a toolkit for developing and comparing algorithms in RL. These environments are used to evaluate the performance of trained agents with algorithms from the Stable Baselines 3 library. This library implements a number of control problems, such as toy problems suited for RL, games of varying levels of complexity as well as many robots and robotic arms that simulate industrial applications [Raffin et al. 2021].

The simulated Industry 4.0 application consists on running the Acrobot-v1 environment, a classic problem in RL. The Acrobot simulates a mechanical arm with two segments and two joints, where the objective is to move the joints so that the lower part of the arm reaches a certain height. In the Acrobot, an episode ends when the robotic arm reaches the target line or when the time limit for reaching the target line is extrapolated. This environment was chosen due to the similarity with Industry 4.0 applications, such as the remote activation of a pick-and-place robotic arm. However, our simulator allows the use of any environment implemented in OpenAI Gym.

[2] `https://man7.org/linux/man-pages/man8/tc-netem.8.html`
[3] `https://github.com/DLR-RM/stable-baselines3`
[4] `https://gym.openai.com/`

We used the Advantage Actor Critic (A2C) algorithm in the controller. A2C is a deep RL algorithm based on value-based methods (e.g, Q-learning, Deep Q-learning) and policy-based methods (e.g., reinforce with policy gradients) [Simonini 2018]. This method uses two neural networks to measure how good the action taken is (value-based), and an actor that controls how the agent behaves (policy-based).

3.3. Evaluation methodology

We defined two evaluation scenarios: one where the agent is trained under ideal network conditions (without impairments between controller and actuator), and another, where the agent's training takes place over the network with diverse impairments inserted. For the experiments with training over the network, agents are trained with the same number of timesteps and the same environment was used to evaluate only the impact of the network impairments on the agent's performance.

Scenario #1: offline training. The first scenario could be used in an AI system that can be trained offline on the cloud. Upon deployment, the actuator is installed on the IoT device and the controller operates in the cloud. This may be the case of different Industry 4.0 applications (e.g, pick-and-place robotic arm control) where the tasks are immutable over time.

Scenario #2: training over the network. The second scenario deals with situations requiring continuous agent training in order to correctly operate under dynamic conditions. One example is a production line for automotive tools, where the parts being built change over time. Only a few units of each part are produced, and then the line takes up another project.

Evaluation metrics. We used the same agents to assess how network quality affects not only the total reward under a ML perspective, but the mean time the agent needs to complete its task. The mean reward is the arithmetic mean of all rewards received after each action taken by the agent during an episode. For each metric, we evaluated the overall agents performance after 100 episodes. For each successful action by the agent, the environment returns a -1 point reward. At the end of the episode, we obtain the full reward for completing it. In the second scenario, we compare each metric with the mean reward obtained with the pre-trained agent. We also calculated a 95% confidence interval of the rewards obtained at the end of each episode.

The graphs in the next section allow us to compare the performance in time and in the number of actions performed for each variation of each QoS metric. In addition, we were able to compare between the metrics when analyzing the data, so that we could infer that for the same pre-trained agent, the network impairments negatively affect its performance.

4. Results

4.1. Agent deployed with prior offline training

Latency: The latency of a network, measured in milliseconds (ms), defines how long a request takes to transfer from one point to another. Without making changes in the network QoS between client and host, an average latency $\leq$ 1ms was observed in the first 10 requests. As expected, smaller rewards can be seen as network latency increases considerably (Fig. 2). Thus, the greater the network latency, the greater the agent's difficulty

in successfully executing the actions, which results in a longer time to achieve the goal (Fig. 3).

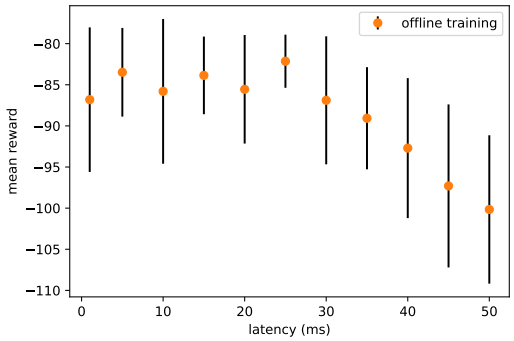

Figure 2. Mean reward by varying network latency with 500,000 timesteps

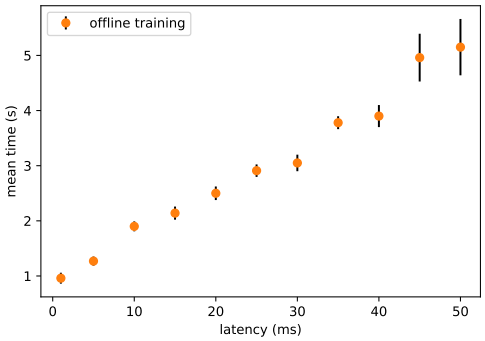

Figure 3. Mean task completion time by varying network latency with 500,000 timesteps

After increasing the delay above 30ms latency, the agents overall performance gets very bad. It is noticed that the mean time in each episode grows considerably as the network latency increases. It also can be seen that the curve drops considerably to a reward of -100 points for Acrobot-v1 environment.

Packet loss: As the packet loss rate increases, timeouts occur in the communication between the agent and the environment. This occurs because packets that indicate the end of an episode can be lost during traffic, causing the agent not to receive the reward for the environment. In these cases, the robotic arm gets static. The simulator then forces the episode to continue by retransmitting the packet, so that the environment still receives it, even if delayed.

With this test, it was possible to verify that with 1% of packet loss, in 94% of the episodes a timeout occurred. Thus, we can see in Figure 4 and Figure 5 the mean reward and mean time respectively obtained by the agent as we increase the packet loss rate.

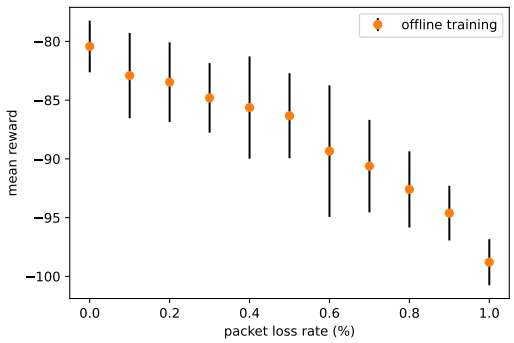

Figure 4. Mean reward by varying network packet loss with 500,000 timesteps

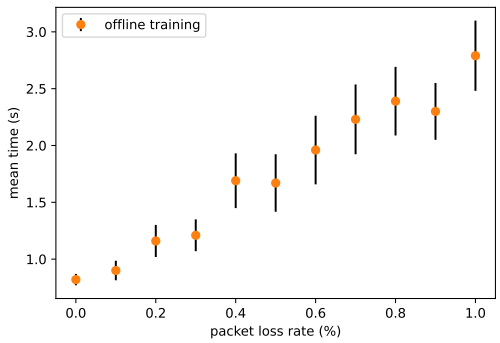

Figure 5. Mean task completion time by varying network packet loss with 500,000 timesteps

Note that the mean time increases considerably due to the number of timeouts

that occur as the network quality deteriorates. Thus, the mean reward of the episodes tends to decrease and become more unstable, which can be noticed by larger confidence intervals. This occurs because the network becomes unstable with a lot of packet losses, and requests are lost. In this way, the agent's behavior gets unstable as well, needing more actions and taking longer to complete the objective. This shows that if some packet containing the information that the episode has reached its end is lost, since the selection of packets that will be lost is arbitrary with NetEm, the application becomes unfeasible for remote control.

Packet corruption: Analogous to other QoS metrics, the worse the network quality, the more difficult it is to remotely control the robotic arm (Figure 7). When evaluating the corruption, an 1% rate returned an average reward of -89.22 points, which is 7.35% worse than the -82.66 points obtained with a corruption rate of 0.1%.

An interesting aspect to note concerns the similarity of the packet corruption and packet loss metrics. Since for the purpose of developing this simulator, a corrupted packet is discarded as well as a lost packet. However, we decided to keep the analysis for both metrics because in a real scenario, AI-based applications can be used in networks with considerable packet corruption rates.

We also can see that while the time to complete an episode increases as the packet corruption rate increases as well, the reward obtained at the end of each episode decreases (Figure 6).

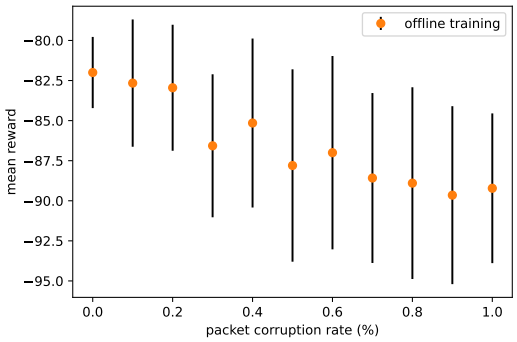

Figure 6. Mean reward by varying network packet corruption with 500,000 timesteps

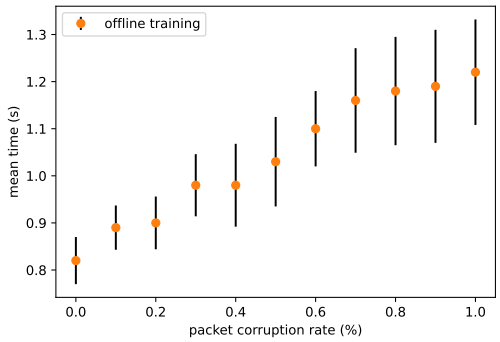

Figure 7. Mean task completion time by varying network packet corruption with 500,000 timesteps

Once the agent resumes executing its actions from the position where it stopped before the timeout exception occurred, the number of actions does not necessarily increase as more timeouts exceptions occur in the network. Therefore, in order to penalize the group of episodes in which more exceptions occurred, we directly related the number of timeouts to the final reward value. In this way, Figure 8 relates the mean reward obtained at the end of the 100 episodes with the number of timeouts exceptions that occurred. Thus, we can see in Figure 9 that by correlating the number of occurrences of timeouts exceptions it is clearly possible to see how the worsening in the network quality affects the overall performance of the agent.

Jitter: Jitter is a statistical variation of the delay in delivering data over a network.

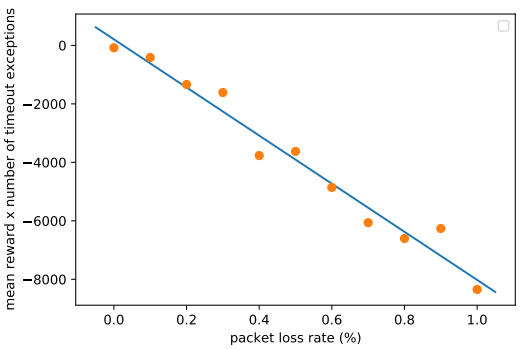

Figure 8. Relation between number of timeouts and mean reward by varying network packet loss with 500,000 timesteps

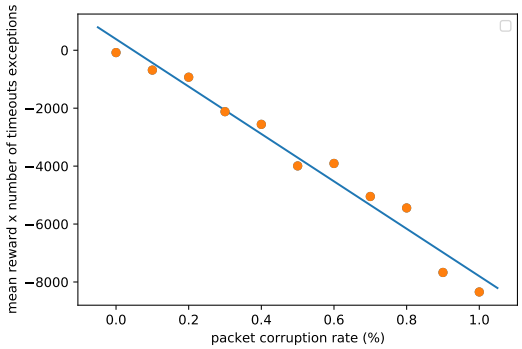

Figure 9. Relation between number of timeouts and mean reward by varying network packet corruption with 500,000 timesteps

Thus, the simulator performs the delay variation between successive data packets in a network. These small variations in latency between the delivery of separate packets can be caused by network congestion, network collisions, or signal interference. By default, the delay parameters are described by the average value (μ) and standard deviation (σ). By default, NetEm uses a uniform distribution, so that the delay is within $\mu \pm \sigma$. For the tests, we considered jitters of 5ms, 10ms, 15ms and 20ms. Therefore, all packets leaving host will experience a delay, with a random variation of jitter. In Figure 10 we can see as the network jitter is 20 ms, the mean time to complete the episode is relatively longer.

Packet reodering: Packet reordering is a well-known phenomenon that the order of packets is inverted inside a network. Using NetEm, it was possible to vary the percentage of packets that suffer delay over network. In this way, we were able to simulate sending some packets through the default average latency $\leq$ 1ms between the VMs and delay sending the others by a few milliseconds. In order to evaluate the agent's behavior based on these statistical variations of packet delivery delay, it was possible to simulate, for example, the immediate sending of 25% of the packets and the others were delayed by 50 ms, which is more like real life, because causes a certain percentage of the packets to get mis-ordered. The test case that most impacted the application's remote control, with the exception of the case where no packet is sent immediately, was sending the first 25% packets first and then delaying the others. Figure 11 shows that as we increase the immediate send rate, the time it takes the agent to reach the goal reduces significantly, because fewer packets will be delayed on the network.

These results shows that the network conditions demand more training of the systems and some QoS metrics affects more the agent behavior than other, like the mean time curve that increases at a relatively greater rate with increasing packet loss than with packet corruption, as the number of actions taken by the agent tends to be similar. When comparing jitter and latency, we can see that the immediate sending rate dictates the curves behavior. As we increased the rate, we could see that the agent's behavior tends to stabilize.

The overall conclusion of this scenario is that some QoS metrics interfere more in the remote control of an application than others. For example, the packet loss rate caused

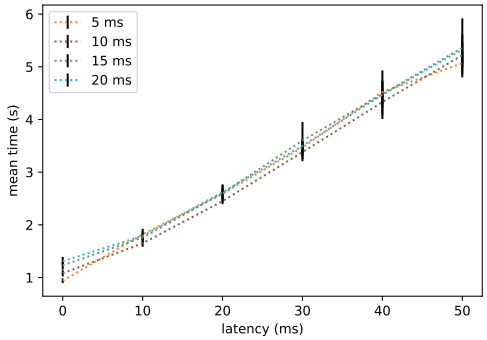

Figure 10. Mean task completion time by varying jitter with 500,000 timesteps

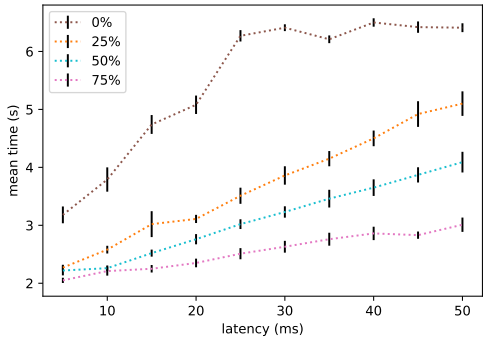

Figure 11. Mean task completion time by varying packet reordering with 500,000 timesteps

the agent's behavior to become inconsistent by taking longer to complete the episode faster than modifying packet corruption rate.

4.2. Agent trained over the network

After analyzing the application-based results when using a pre-trained agent with 500,000 timesteps, the test scenario can be extended. It was possible to observe the agent's behavior taking into account the training process. For this, we use an online trained agent under the network parameters modifications. The aim is to see how the agent behaves when its training is susceptible to network quality, such as delays and packet losses. We could see that network conditions demand more training from agents, since more robust training is needed in a network with poorer quality. Thus, the offline trained agent's performance in the same OpenAI Gym's environment with 20,000 timesteps was compared with an online trained agent's performance.

Latency: Figure 13 displays the comparison between the pre-trained agent (offline) and the agent trained over network conditions (online) when varying latency under required metrics to a power grid system in industry [Nasrallah et al. 2018]. It can be seen that the first agent performed relatively better than the second, since the quality of the network did not influenced its training. Even though both have been trained on the same algorithm and with the same number of training timesteps, the agent with offline training has a less disparate time curve, while the online trained agent was susceptible to performance deterioration. Figure 12 shows the increase in the number of actions performed within an episode and the decrease in its mean reward.

Packet loss: When considering the packet loss rate of a real network, it was possible to evaluate the behavior of the online trained agent. In order to compare the online trained agent with the pre-trained agent, Figure 15 exposes how the mean time for the robotic arm to reach the target increases considerably as the network quality deteriorates at higher rates for the agent with online training. The mean reward obtained by the agent deployed with zero training drops as the packet loss rate exceeds 0.5%, and the rewards obtained are always worse than the mean reward obtained by the agent with previous training (Fig. 14).

It can be seen that the network latency for the agent deployed with zero training

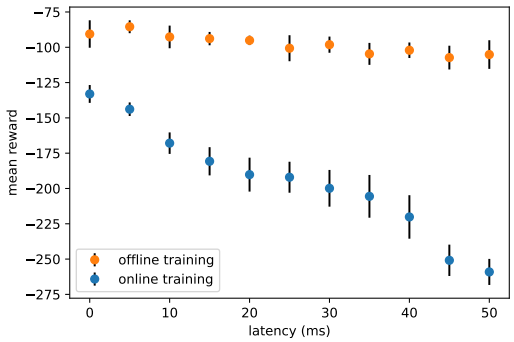

Figure 12. Mean reward by varying network latency with 20,000 timesteps

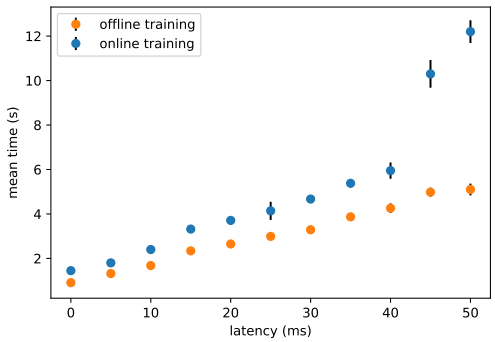

Figure 13. Mean task completion time by varying network latency with 20,000 timesteps

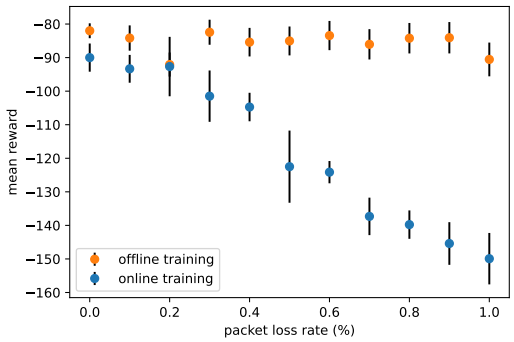

Figure 14. Mean reward by varying network packet loss with 20,000 timesteps

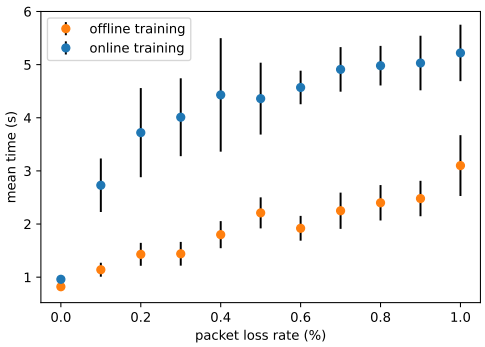

Figure 15. Mean task completion time for varying packet loss on the network with 20,000 timesteps

causes the reward to drop to considerably lower levels in the first latency variations than with the first increases in the packet loss rate. When comparing the two metrics side by side, it is noticeable that the time to complete an episode is longer as the latency increases. Above all, the mean reward obtained at the end of the episode when the rate approaches 1% of packet loss drops substantially.

Packet corruption: Finally, when taking into account the packet corruption rate, it is possible to analyze a notorious lack of control in the behavior of the robotic arm in the episodes in which the corruption rate approached 1%. In these situations, the fact that the online trained agent learns to perform his actions under a network in which the packets related to his training were corrupted, made him take longer to reach his objective. Figure 17 presents the mean time to complete the episode, which approaches 3 seconds. Thereby, we can see how an online trained agent's rewards is always lower than the mean reward obtained with the offline trained agent. Also note that the rewards obtained from the pre-trained agent remains even with changes in network quality. As we use a pre-trained agent under ideal network conditions, it is natural that your training will maintain the same agent's performance, as we can see in Figure 16.

The results above indicate that is better to use a pre-trained agent that has been

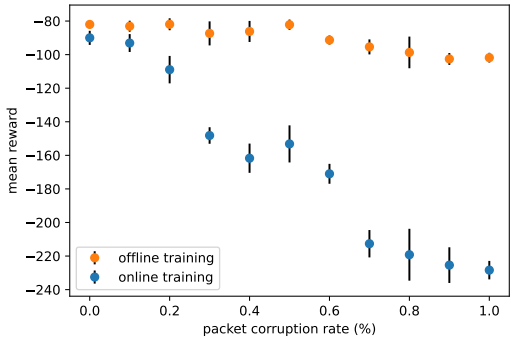

Figure 16. Mean reward by varying network packet corruption with 20,000 timesteps

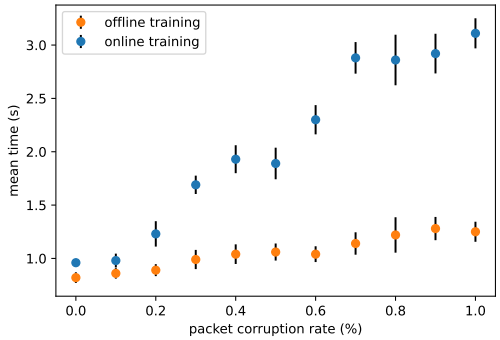

Figure 17. Mean task completion time for varying network packet corruption with 20,000 timesteps

trained under ideal network conditions than deploying an agent without any training in an unstable network. This paper evaluates static network conditions, since it is a first step towards more complex scenarios. Future work will consider dynamic networks with time-varying conditions.

5. Conclusions and future work

The present work allowed us to evaluate the results of the effects of QoS network characteristics on the performance of simulated remotely controlled applications in Industry 4.0. By remotely controlling an industrial device, the operator is able to maintain a real-time bidirectional interaction. Focusing on the interaction with industrial processes, some examples were presented to show the existing resources that are already applied in the industry. With the integration of simulation tools for networks and physical processes, a simulator was developed that can measure the impact of variations in the QoS in a network on the performance of the actions by an agent in a simulated environment. Furthermore, it was possible to evaluate the results not only by taking into account the application, but also the agent's own online training.

For future work, we will take into account the use of the simulator to detect when a network experiences losses and delays and dynamically train the agent to increase the reward obtained. The RL concept can be extended by using different QoS metrics together and testing more scenarios. The biggest motivation for this study is that AI is becoming the future of everything we know as technology today. It is extremely necessary that we understand the impact of new networks and URLLC applications as well as observe which QoS metrics have the greatest effect on the control of these applications. AI needs to dynamically adapt to the environment, with agents that create alternatives to circumvent these instabilities in a real network, such as losses and delays.

Acknowledgment

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq (funding agency from the Brazilian federal government), FAPEMIG (Minas Gerais State Funding Agency), and São Paulo Research Foundation (FAPESP) with Brazilian Internet Steering Committee (CGI.br), grants 2018/23097-3 and 2020/05182-3.

References

Akpakwu, G. A., Silva, B. J., Hancke, G. P., and Abu-Mahfouz, A. M. (2017). A survey on 5g networks for the internet of things: Communication technologies and challenges. *IEEE access*, 6:3619–3647.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Beltrão, S. R. B. B. and de Oliveira Pires, A. A. (2019). Generation of corporate intelligence in industry 4.0, through the combination of professional drones and artificial intelligence. case study applied to a coal recovery boiler. *Brazilian Journal of Technology*, 2(4):946–966.

Buchner, R., Wurhofer, D., Weiss, A., and Tscheligi, M. (2012). User experience of industrial robots over time. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 115–116.

Chen, Y., Farley, T., and Ye, N. (2004). Qos requirements of network applications on the internet. *Information Knowledge Systems Management*, 4(1):55–76.

Cheng, J., Chen, W., Tao, F., and Lin, C.-L. (2018). Industrial iot in 5g environment towards smart manufacturing. *Journal of Industrial Information Integration*, 10:10–19.

Civerchia, F., Giannone, F., Kondepu, K., Castoldi, P., Valcarenghi, L., Bragagnini, A., Gatti, F., Napolitano, A., and Borromeo, J. C. (2020). Remote control of a robot rover combining 5g, ai, and gpu image processing at the edge. In *Optical Fiber Communication Conference*, pages M3Z–10. Optical Society of America.

ETSI (2020). Service requirements for cyber-physical control applications in vertical domains. Acessed in 01.10.2022.

Faheem, M., Shah, S., Butt, R., Raza, B., Anwar, M., Ashraf, M., Ngadi, M., and Gungor, V. (2018). Smart grid communication and information technologies in the perspective of industry 4.0: Opportunities and challenges. *Computer Science Review*, 30:1–30.

Harris, C. J. (1994). *Advances in intelligent control*. CRC Press.

Huang, G., Rao, P. S., Wu, M.-H., Qian, X., Nof, S. Y., Ramani, K., and Quinn, A. J. (2020). Vipo: Spatial-visual programming with functions for robot-iot workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Jiang, Z., Fu, S., Zhou, S., Niu, Z., Zhang, S., and Xu, S. (2020). Ai-assisted low information latency wireless networking. *IEEE Wireless Communications*, 27(1):108–115.

Kunst, R., Avila, L., Binotto, A., Pignaton, E., Bampi, S., and Rochol, J. (2019). Improving devices communication in industry 4.0 wireless networks. *Engineering Applications of Artificial Intelligence*, 83:1–12.

Liberato, A., Martinello, M., Gomes, R. L., Beldachi, A. F., Salas, E., Villaca, R., Ribeiro, M. R., Kondepu, K., Kanellos, G., Nejabati, R., et al. (2018). Rdna: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters. *IEEE Transactions on Network and Service Management*, 15(4):1473–1487.

Liu, Q., Zoppi, S., Tan, G., Kellerer, W., and Steinbach, E. (2017). Quality-of-control-driven uplink scheduling for networked control systems running over 5g communication networks. In *2017 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, pages 1–6.

Lobbrecht, A. H. and Solomatine, D. P. (2002). Machine learning in real-time control of water systems. *Urban Water*, 4(3):283–289.

Luo, G., Yuan, Q., Li, J., Wang, S., and Yang, F. (2021). Artificial intelligence powered mobile networks: From cognition to decision. *arXiv preprint arXiv:2112.04263*.

Marques, P., do Carmo, A. P., Frascolla, V., Silva, C., Sena, E. D., Braga, R., Pinheiro, J., Astudillo, C. A., de Andrade, T. P., Gama, E. S., et al. (2018). Optical and wireless network convergence in 5g systems–an experimental approach. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–5. IEEE.

Nakimuli, W., Garcia-Reinoso, J., Sierra-Garcia, J. E., Serrano, P., and Fernández, I. Q. (2021). Deployment and evaluation of an industry 4.0 use case over 5g. *IEEE Communications Magazine*, 59(7):14–20.

Nasrallah, A., Thyagaturu, A. S., Alharbi, Z., Wang, C., Shao, X., Reisslein, M., and ElBakoury, H. (2018). Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research. *IEEE Communications Surveys & Tutorials*, 21(1):88–145.

Papcun, P., Kajáti, E., and Koziorek, J. (2018). Human machine interface in concept of industry 4.0. In *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pages 289–296.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Rodriguez, I., Mogensen, R. S., Fink, A., Raunholt, T., Markussen, S., Christensen, P. H., Berardinelli, G., Mogensen, P., Schou, C., and Madsen, O. (2021). An experimental framework for 5g wireless system integration into industry 4.0 applications. *Energies*, 14(15):4444.

Simonini, T. (2018). An intro to advantage actor critic methods: let's play sonic the hedgehog! https://www.freecodecamp.org/news/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d/. Acessed in 06.02.2022.

Yang, S.-Y., Jin, S.-M., and Kwon, S.-K. (2008). Remote control system of industrial field robot. In *2008 6th IEEE International Conference on Industrial Informatics*, pages 442–447. IEEE.