Real-Time Data Transmission Optimization on 5G Remote-Controlled Units Using Deep Reinforcement Learning

Nikita Smirnov(✉) and Sven Tomforde

University of Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, Germany
{nsm,st}@informatik.uni-kiel.de
https://www.ins.informatik.uni-kiel.de/

Abstract. The increasing demand for real-time data transmission for the remote-controlled units and the complexity of 5G networks pose significant challenges to achieving optimal performance in device-based scenarios, when the 5G network cannot be controlled by its users. This paper proposes a model-free Deep Reinforcement Learning approach for this task. The model learns an optimal policy for maximizing the data transmission rate while minimizing the latency and packet loss. Such an approach aims to investigate the applicability of the environment-agnostic agents driven purely by the transmission statistics of the acknowledged packets. The evaluation is done with the help of a 5G simulation based on the OMNeT++ network simulator and the obtained results are compared to a classic throughput-based adaptive bitrate streaming approach. Multiple questions and challenges that arose on the way to the final model and evaluation procedure are highlighted in detail. The resulting findings demonstrate the effectiveness of Deep Reinforcement Learning for optimizing real-time data transmission in 5G networks in an online manner.

Keywords: deep reinforcement learning · data transmission · adaptive bitrate streaming · 5G networks · remote-controlled unit · organic computing

1 Introduction

Organic Computing (OC) is a research field that aims to maintain the controllability of technical systems in the face of ever-increasing complexity by shifting tasks from the developer to the system itself [10]. The result is typically collectives of self-adaptive and self-organizing systems using machine learning technology that makes independent decisions based on objective functions. Especially learning technology from the field of Deep Learning (DL) and Deep Reinforcement Learning (DRL) has been proven as key-enablers for OC-capabilities.

A highly topical area of application for OC technology is autonomous systems, such as those found in the context of autonomous shipping. In this paper,

G. Goumas et al. (Eds.): ARCS 2023, LNCS 13949, pp. 281–295, 2023.
https://doi.org/10.1007/978-3-031-42785-5_19

we consider the scenario of an autonomous ferry navigating within the overloaded maritime areas, see e.g. [19]. As full autonomy is currently restricted due to legal issues, either an onboard or a remote control by a human is required. As part of current research projects, the unmanned ferry (see Fig. 1) is equipped with fifth-generation (5G) wireless communication technology to transfer its sensor data such as video and 3D point cloud flows to the shore-based remote control center.

The rapid development of 5G technology has increased the potential reliability of remote-controlled units (RCUs). The real-time data transmission in 5G-based RCUs is crucial for ensuring seamless communication between the operator and the unit, which demands low latency, high reliability, and efficient use of network resources. Since in our ferry scenario, dramatically more data is generated than 5G capacity is available, the system has to select and adapt the communicated data at runtime based on changing conditions.

The main contribution of this paper is the application, setup and testing of a general-purpose Deep Reinforcement Learning (DRL) for optimizing the real-time data transmission on the RCUs in 5G networks, where DRL agent learns to make optimal decisions based on the network conditions while so far avoiding using 5G environmental data and any knowledge about the problem. A simulation-based testbed is developed to verify the proposed approach while using both simulated and real data. This paper contributes to the growing body of research on OC and the application of DRL to wireless communication networks and it provides insights into the potential benefits of using DRL solutions in 5G networks or OC systems in general.

This paper is organized as follows: Sect. 2 shortly reviews the related work, Sect. 3 presents the optimization problem, Sect. 4 describes the proposed approach, Sect. 5 addresses evaluation and results and Sect. 6 briefly recaps the content of previous parts and concludes the paper with possible future work.

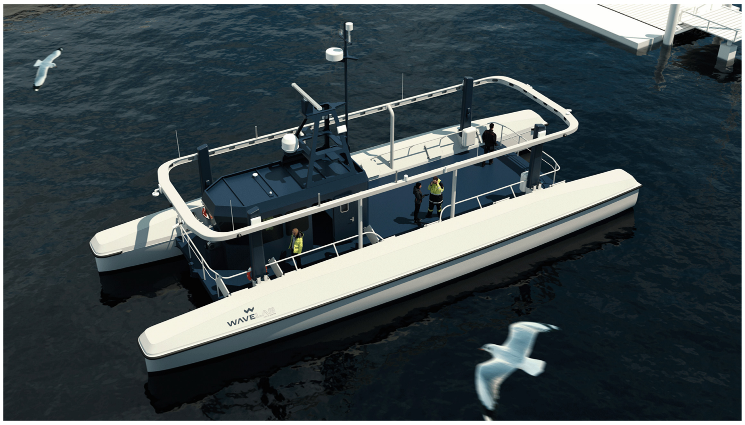


Fig. 1. Design of a 5G remote-controlled passenger ferry “Wavelab” for the Bay of Kiel.

2 Related Work

Considering optimization problems in 5G networks with machine and/or deep learning applications, there is a huge trend towards network-based solutions [14]. Unlike device-based problems, it is assumed that a developer has partial or even full control over a 5G network and may change the hardware or software parts of it so that optimization problems are more concentrated on the network-management aspects rather than on the data transmission ones. Typical examples of such problems could be network slicing, power allocation and control, scheduling, handover management, etc. [16].

A standard approach to optimize data transmission is known as adaptive bitrate streaming (ABR). Current edge- and cloud-solutions are very diverse and are massively used in everyday streaming, especially by popular video streaming services like YouTube, Netflix and other popular services. Standard techniques are based on controlling two main features: a) video bitrate, i.e., the amount of data send in the current period and b) playback stability, i.e., how smooth and continuous is the video playing on the consumer side [3].

Intelligent solutions enhance standard techniques with smart retransmission mechanisms [12], with DL-based predictive assistance [1] and also with DRL integration by introducing an agent with a state including bitrate, downloading time of the previous stream chunks and buffer occupancy [2,8]. Almost all of the existing approaches for adaptive bitrate streaming assume that the data is **on-demand** and is transmitted over the popular contemporary streaming protocols like MPEG-DASH, HLS, WebRTC and others [7] so that they try to optimize both throughput and playback smoothness. A pure bitrate-based approach with DRL assistance is used in [15] without directly taking into account latency and packet loss factors, which are very important in 5G networks.

Another important factor is that the sensors on remote-controlled units are usually heterogeneous, not only cameras but often other remote sensing devices like LiDARs and RADARs that produce 3D point clouds. Although there is a way how to reduce point clouds to a video flow by using classic MPEG video encoders as a compression tool [18], the most popular approach still involves space-partitioning trees [4], so that a generalization of adaptive bitrate streaming solutions for all possible types of data with their encoding options is needed.

3 Real-Time Data Transmission Problem

Improving performance for real-time data transmission is a more complicated and challenging task in comparison to a standard adaptive bitrate streaming since the consumer's buffer cannot be filled for some time in advance, otherwise, a human operator of the controlled unit will deal with outdated data and playback delays. Therefore, the only way is to try to optimize the bitrate directly. Due to this constraint, it is not possible to use standard HTTP algorithms like throughput-based FESTIVE [6] or buffer-based BOLA [20], since they all presuppose that: a) the consumer's buffer is used and could be manipulated, b) it is possible to have simultaneous streams with different bitrate presets. While the second condition can be so far neglected since the data, especially video data, can be encoded on-the-fly relatively fast, the first one remains unfeasible.

Given the above, the optimization problem designated in this article as "real-time data transmission" could be formulated as follows:

> try to **maximize the uplink goodput (GPT)** while at the same time **minimize the round-trip-time (RTT)** and **minimize the packet loss rate (PLR)** *by* selecting a proper bitrate for one or multiple streams for a certain period *without* holding the data in some buffer neither on the sender nor the receiver side.

It is assumed that goodput, as well as RTT, are measured at the application level, and the processing delays for unpacking and decoding the data are neglected. It is also assumed that **all optimization happens directly on the RCU**, therefore, there is no information about the packet as long as the receiver's acknowledgment does not come back. A unit uses "best-effort" transport protocols (UDP-based ones) since retransmission mechanisms are not required.

There is no ready-made formula for selecting the optimal bitrate for the next period based on the previously used bitrate and collected network feedback. *It may be even unfeasible in general due to the complexity of relations between these parameters.* The only known connection between throughput, RTT and PLR is Mathis formula for TCP congestion algorithm [9]:

$$\mathrm{BW} < \frac{\mathrm{MSS}}{\mathrm{RTT}}\frac{1}{\sqrt{\mathrm{p}}} \tag{1}$$

where BW is a bandwidth of a TCP connection, MSS is the maximum segment size (usually equals to 1460 bytes considering IP MTU of 1500 bytes) and p is a probability of a packet loss. This formula only gives an upper bound in realistic scenarios, requires a predictable loss rate and is tailored for a TCP congestion window.

There is a wide range of interconnected factors that can impact the overall performance of the data transmission and the network feedback and not all of them could be controlled by a unit itself. For example, RTT can vary depending on the uplink or downlink congestion induced by other 5G devices as well as the number of network hops in between, while PLR is strongly influenced by the distance and the orientation towards the nearest 5G base stations and physical channel quality at the moment. And since all optimization happens on a unit and a public 5G network is used, the middle layers of the core network as well as the intern computations of the base stations are hidden. Temporal relations between GPT, RTT and PLR could be studied only by observing the transmission behavior.

In summary, the absence even of an approximate formula in the general case that describes complex relationships between the key parameters and the overall complexity of a problem was the initial motivation to try to learn it with a model-free DRL approach: to investigate how far the agent could discover these relations, how well it responds to their changing behavior and can optimize its own decisions based on the observed experience of real-time data transmission in the 5G network.

4 Approach: Deep Reinforcement Learning

As mentioned above, model-free DRL was chosen as the main approach in this paper. The main advantage in comparison to the supervised learning is that it does not require any labeled data or the expert knowledge, which are both hard to be obtained for this problem. As a disadvantage, the results are dependent on the manual fine-tuning of agent's behavior control encoded through the reward function. It was also so far consciously decided not to use any device-available 5G information to help to train the agent to investigate the limits of applicability of **environment-agnostic feedback-driven approach**, i.e., the agent learns so far independently from the actual communication technology, location, etc. so that the achieved knowledge is easier to be transferred to other scenarios.

4.1 Background and Model

Reinforcement Learning is a type of machine learning that involves training an agent to learn through interaction with the environment. The agent receives feedback in the form of rewards or penalties for each action it takes. Markov Decision Process (MDP) is a mathematical framework used to model decision-making problems. It assumes that the current state of the environment contains all relevant information necessary to make a decision and that the environment follows the so-called Markov property, meaning that the process is memoryless: the probability of selecting the next state depends only on the current one. DRL is a further development of this strategy that utilizes deep neural networks (DNN) to approximate the agent's policy and value functions in MDP, see Fig. 2.

Proximal Policy Optimization (PPO) [17] was selected as a primary model among different contemporary DRL algorithms. It is an on-policy algorithm that employs a clipped surrogate objective function that constrains policy updates to prevent large policy changes that could negatively impact learning stability. One

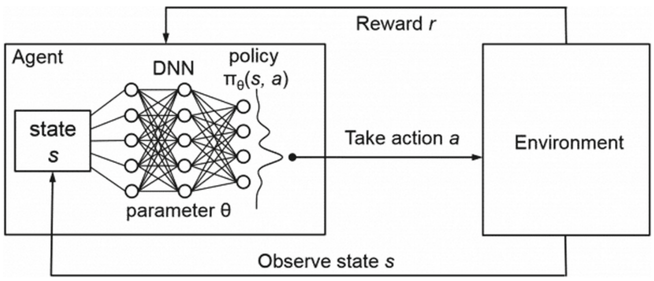


Fig. 2. Deep Reinforcement Learning workflow.

big advantage is its ability to handle both discrete and continuous action spaces. Furthermore, PPO is commonly used not only with fully-connected DNNs but also with Long Short-Term Memory (LSTM) layers, which could increase the ability to learn complicated temporal relations in sequential data and opens a perspective for further improvements.

4.2 Setup and Hyperparameters

Real-time data transmission could be always formulated as a single-agent problem by fixing the interval between two consecutive actions (transition). If a remote-controlled unit has multiple sensors and consequently multiple simultaneous streams, they could be seen as a single flow, where the current common bitrate is spread between the streams either equally or according to some prioritization scheme. It was found after some experiments that the interval equal to one second provides the best trade-off between a) a sufficient amount of sent packets together with received transmission feedback and b) a sufficiently frequent reaction of the DRL model. The first point eliminates a credit assignment problem, which would inevitably arise if the next action is applied too early, even maybe before the acknowledgments for the previous packets are delivered through a congested network. The second point provides enough flexibility for the DRL agent to control the transmission process.

The only negative drawback of this approach is the time it takes to train the agent. Since the agent waits some time to observe enough transmission with selected action, the training cannot be accelerated with modern GPUs. Even in the simulated environment, it is needed around 20–30 s to train the agent for 100 steps with one second transition period because a simulation consumes only CPU resources. On the other hand, it could be easily parallelized horizontally by training several different DRL setups simultaneously on a multi-CPU system. This drawback additionally highlights the importance of fine-tuning the DRL setup.

State. A state represents the most essential agent's parameters and the current agent's perception of the data transmission. It contains both values for a current period as well as global average values for the whole run:

$$S = \{G_p^{rx}, G_g^{rx}, R_p, R_g, J_p, J_g, P_p, P_g, V_p, Q\} \qquad (2)$$

where low indexes denote either a value for the current period, p or the global average g, and the upper index rx means the receiver side. G means goodput, R - round-trip-time, J - jitter, P - packet loss rate, V is the relative standard deviation of the chunks, i.e., how much is the difference between the sizes of generated chunks within a period and Q means the quality stability, it measures how often the bitrate has been changed over the previous ten steps.

Action Space. Action space in this article is discrete and fixed so far to a size of six to compare it with standard ABR solutions: five values correspond to meaningful bitrate values and the last action turns the transmission completely off, it is the "last hope action". A set of bitrate values in kilobits per second could look like that for one video stream: $\{800, 1500, 3000, 6500, 10000\}$ reflecting typical presets from SD to 4K. It should be noted that real-world encoders cannot guarantee that they will precisely fit into the set bitrate value, there are always oscillations. This peculiarity is also transferred to the simulated streams, they may exceed with some chance an average packet size assumed for that bitrate.

Reward Function. Designing a reward function (RF) was the most challenging task due to the complex and dynamic nature of the network environment. RF needs to balance the competing objectives, such as maximizing GPT and minimizing RTT and PLR. Additionally, it should be robust to changes in network conditions and should adapt to new situations and scenarios. The function takes the current state as an argument and consists of three subparts, each of which is "responsible" for regulating one of the main optimization targets (GPT, RTT, PLR):

$$R(S) = 0.33R_{gpt}(S) + 0.33R_{rtt}(S) + 0.33R_{plr}(S) \tag{3}$$

Below is an explanation of all the parts presented in Eq. 3:

$$\begin{aligned} &R_{gpt} = 0.33\frac{G_p^{rx}}{G_{max}} + 0.33\frac{G_g^{rx}}{G_{max}} + 0.33\min(\frac{G_p^{rx}}{G_g^{rx}}, 1) \\ &0.5 \leq V_p < 0.75 \rightarrow R_{gpt} = 0.95R_{gpt}, V_p \geq 0.75 \rightarrow R_{gpt} = 0.9R_{gpt} \\ &R_{gpt} = R_{gpt}(1 - \frac{Q}{10}), \end{aligned} \tag{4}$$

R_{gpt} in Eq. 4 is a combination of three parts, first two examine a current global and periodic goodput at the receiver side in comparison with the maximum possible, and the last part estimates how good the agent *intents* not to perform worse than it already has. Having all three parts allow to control the goodput from all the sides and adjusting weights 0.33 were found to be the most optimal during the training. Two additional regulative mechanisms reduce the price for a goodput if a relative standard deviation of packet sizes is too big (compensates a key-frame difference) and if a bitrate is switched too often.

$$\begin{aligned} R_{rtt} =: &\begin{cases} 1 - 5R_p & 0 < R_p \leq 0.1 \\ 0.6 - R_p & 0.1 < R_p \leq 0.5 \\ 0 & 0.5 < R_p < 0.9 \\ -1 & \text{otherwise} \end{cases} \\ &J_p \geq 0.1 \land R_{rtt} > 0 \rightarrow R_{rtt} = 0.9R_{rtt} \end{aligned} \tag{5}$$

R_{rtt} in Eq. 5 is built on the top of "expert knowledge". Specifications for RTTs are very tight to maintain reliability: zero is given for the RTT below half a second, and for more than 0.9 it is a constant maximum punishment. The idea is to give an extremely high reward for RTT below 100 ms., proportional reward for the RTT that is considered to be enough to provide a "good" quality of experience and harshly punish otherwise. The additional punishment is added for having a too-big jitter.

$$\begin{aligned} &R_{plr} = 1 - 2P_p \\ &\text{if no packets received} \rightarrow R = -0.25 \end{aligned} \tag{6}$$

R_{plr} in Eq. 6 is a linear function over a PLR, which is very vulnerable to a changed loss rate allowing to quickly punish in case of big packet drops. The last condition controls a case, when there are no packets received. It could be due to the turned-off stream as well as due to extreme congestion. It also softens the punishment from the RTT part to encourage the agent to select lower qualities.

Finally, RF is clipped in the interval $[-1, 1]$ and is equally weighted over each of the three subparts. This makes it easier to analyze and interpret a cumulative reward over multiple episodes as well as to analyze a reward distribution itself.

Hyperparameters. Hyperparameters were adjusted during multiple experiments. The main idea was to encourage the model for more exploration while still keeping a good balance. Therefore values for learning rate, entropy and clip range are changed from default ones. The size of one update was chosen to be equal to the number of steps in a single episode, the latter was chosen to be long enough (more than 10 min of data transmission) while also divisible by a batch size of 64. Network architecture is the same for both actor and critic with two hidden layers each consisting of 128 neurons. The PPO implementation was taken from stable-baselines3 library [13]. The full list of hyperparameters and the default values are presented in Table 1.

Table 1. Selected hyperparameters for the PPO model in comparison to default values.

Hyperparameter	Value	Default Value
Episodes	100	10
Steps	640	2048
Batch Size	64	256
Learning Rate	0.00025	0.0003
Gamma (discount factor)	0.99	0.99
GAE λ	0.95	0.95
Clip Range	0.1	0.2
Entropy Coef	0.01	0.0
Value Coef	0.5	0.5
Max Grad Norm	0.5	0.5
Policy Network Architecture	MLP (2 layers, 128 units each)	MLP (2 layers, 64 units each)
Value Network Architecture	MLP (2 layers, 128 units each)	MLP (2 layers, 64 units each)

4.3 Challenges

To overcome the survivorship bias, this sub-section presents the main challenges that arose on the way to the final design resulting also in rejected DRL setups that showed worse results. Some are caused by the environment, and some by the DRL setup itself:

– Stochasticity of the environment. The 5G network itself is very volatile, the next state of the DRL agent might depend not only on its actions but also on how the network behaves as a whole. That is why it has been decided to start with simulations - to minimize this factor at the beginning considering the more predictable behavior of other 5G devices.
– Low actions variation, i.e., the agent doesn't show an adaptation. It might be stuck with the maximum quality even facing extreme congestion with the hope of compensating current punishments with forthcoming rewards for maximizing a goodput or it might be too careful by not increasing the quality when the network is free. The reward function presented in this paper tries to neutralize this problem.
– "Irrational behavior" due to the cumulative nature of DRL. The agent learns a policy to maximize the cumulative reward over an episode, so it elaborates a strong bias towards certain actions or strategies, which may seem not optimal in current circumstances and may be even considered "irrational" by a human expert. An underrepresentation of "good" or "bad" network situations during the training also belongs to this problem.
– Downlink congestion. RCU waits until the package makes the full round to register it as "received" and update the statistic. If there is strong congestion in the downlink direction, the agent may misinterpret the quality of current data transmission.

5 Evaluation

5.1 5G Simulation and Scenario

The evaluation of the presented approach was conducted by developing a 5G playground using the open-source Simu5G library: the OMNeT++-based 3GPP-compliant 5G simulation written in C++ [11]. Additional modules were developed to enable working with real video and point cloud data together with encoding/decoding on-the-fly. Another implemented feature is the inter-process communication of a simulation in C++ with DRL modules from stable-baselines3 library in Python [13]. Such an approach allows to bring together multiple advantages: a) use OMNeT++ discrete events and message scheduling infrastructure, b) leverage well-developed python DL-stack, c) ease further transfer to the real RCUs, where the inter-process communication with C++-based simulation will be replaced with the external communication with the real world, leaving all other parts unchanged.

The main scenario represents a sandbox, where a unit transmits the data to a server within the 5G network. The setup is adapted to a maritime field: a unit is a ferry, which sends the data from its cameras and LiDAR sensors, and a server is a shore-based station. The other 5G devices are spread over the base stations. There are also background cells imitating 5G base stations from other providers in the area, producing signal interference and noise. A full run configuration is given in Table 2, a schematic illustration of a sandbox is given in Fig. 3.

The idea of the presented sandbox scenario is to train the DRL agent with different network experiences from almost perfect to loss rates of over 70% by increasing the number of devices and their data rates as well as selecting the areas, where the agent is moving during the run. If it moves towards the bottom and right-bottom areas, then it suffers from strong congestion. On the contrary, it enjoys perfect 5G coverage, minimal delays and scheduling priority in the top and top-left areas.

Simulated streams were used during the training of the DRL model to speed up the process. Apart from saving time on reading and visualizing, they differ from the real data only by skipping the encoding, the bitrate is regulated numerically. However, the real video and LiDAR data are often used during the testing phase to verify the transmission "tolerance level" through the visualization. Sometimes "bad" transmission may be even acceptable if it still allows a human operator to adequately perceive the situation. For example, the artifacts of lost frames could be more tolerable than prolonged stalling. This process is illustrated in Fig. 4.

Table 2. A full configuration for the "sandbox" 5G scenario for DRL training.

Parameter	Value	Description
Number of gNBs	2	Number of 5G base stations (gNBs)
Number of BgCells	3	Number of background cells (BgCells)
Number of UEs	80–130	Number of other 5G devices (UEs)
Mobility Model	Random waypoints	Random trajectories for UEs
Playground area	1.5 km^2	The area for a playground
UL traffic main UE	$[2.5, 20]$ mbit/s	Uplink traffic range for a DRL-controlled device
UL traffic other UEs	$[4, 8]$ mbit/s	Uplink traffic range for other UEs
DL traffic other UEs	$[2, 6]$ mbit/s	Downlink traffic range for other UEs
Simulation time	640 sec	Maximum length of one simulation run in seconds
Carrier frequency	3.6 GHz	Frequency used for transmission and reception
Bandwidth	100 MHz	Amount of frequency spectrum
Scheduling discipline	MAX C/I	Allocate resources according to signal-to-noise ratio
Antenna configuration	2×2 MIMO	Antenna configuration
NR scenario	Urban macro-cellular	3GPP-based 5G scenario for outdoor urban areas

The repeating OMNeT++ seeds were used to provide a reproducibility of a simulation process. The maximum bitrate was set to 10 mbit/s and constant bitrate mode (CBR) was turned on. However, PPO model itself is stochastic and always estimates actions' probability and then samples accordingly. To overcome this, it is instructed to take the *argmax* on each step. It does not make a

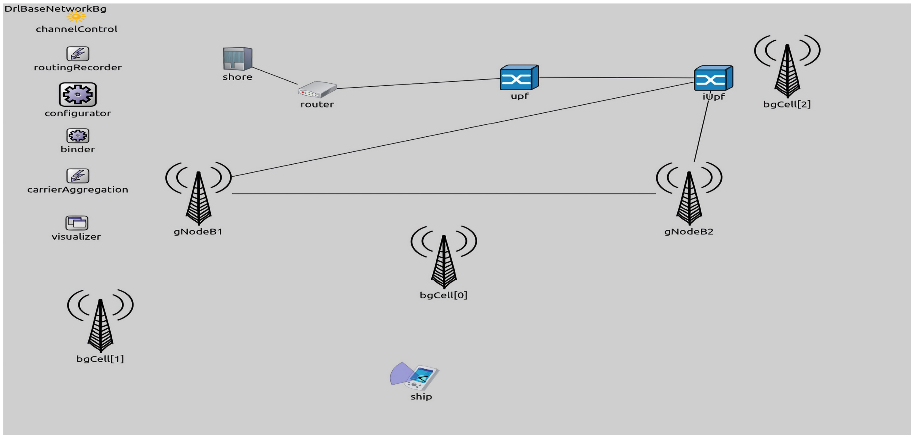
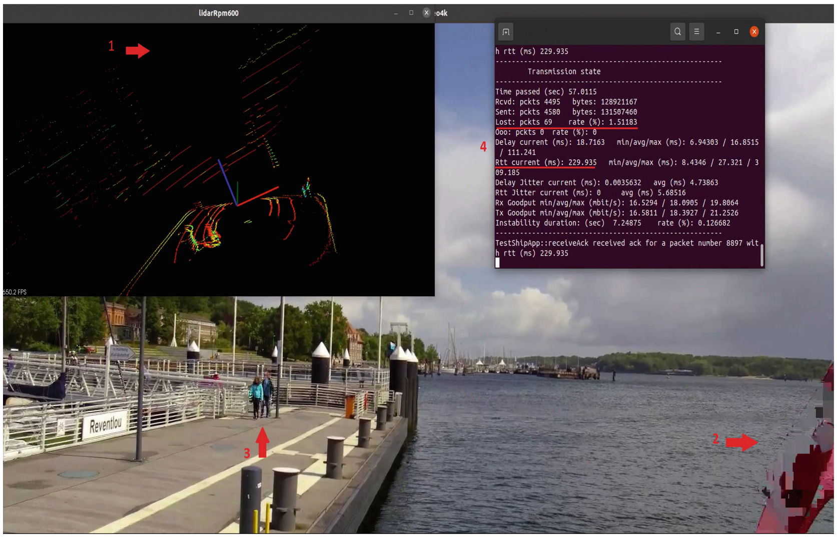


Fig. 3. OMNeT++ visualization of the "sandbox" scenario.

learned policy deterministic, on the contrary, it shrinks the variability of possible decisions, which often results in worse predictions [5], however for the first proof-of-concept it was considered an acceptable reduction.

5.2 Results

The analysis of obtained results consists of several stages. First, the training results are analyzed. Next, the agent is tested on three unseen validation sce-



```
h rtt (ms) 229.935
-------------------------------------------------------
        Transmission state
-------------------------------------------------------
Time passed (sec) 57.0115
Rcvd: pckts 4495   bytes: 128921167
Sent: pckts 4580   bytes: 131507460
Lost: pckts 69    rate (%): 1.51183
Ooo: pckts 0  rate (%): 0
Delay current (ms): 18.7163   min/avg/max (ms): 6.94303 / 16.8515
 / 111.241
Rtt current (ms): 229.935   min/avg/max (ms): 8.4346 / 27.321 / 3
09.185
Delay Jitter current (ms): 0.0035632   avg (ms) 4.73863
Rtt Jitter current (ms): 0     avg (ms) 5.68516
Rx Goodput min/avg/max (mbit/s): 16.5294 / 18.0905 / 19.8064
Tx Goodput min/avg/max (mbit/s): 16.5811 / 18.3927 / 21.2526
Instability duration: (sec)  7.24875    rate (%): 0.126682
-------------------------------------------------------
TestShipApp::receiveAck received ack for a packet number 8897 wit
h rtt (ms) 229.935
```

Fig. 4. Visualization of a test run with one video and one LiDAR stream. The data is taken from a regular ferry and loaded into a simulation. **1** illustrates a segment loss in LiDAR stream, **2** demonstrates an artifact, **3** is a blurring effect originating from a too-big delay. One can see on **4** a console with the current transmission statistic.

narios roughly labeled as “easy”, “moderate” and “hard” depending on the area and workload. Finally, its performance is compared with the simple throughput-based ABR algorithm, which selects the next bitrate by analyzing the goodput of the previous step. If it is stable for three steps in a row, it selects the next available higher quality until the maximum one is reached and downgrades it immediately in case the goodput is reduced. The idea is to compare: a) a DRL solution with an ABR one, and b) a model-free DRL approach with a rule-based one.

Figure 5 illustrates a mean episodic reward (MER) collected during the training. It is a classic learning curve: first MER explodes exponentially, then decreases a bit and finally stabilizes on some plateau. As stated in Sect. 4.2, the maximum step reward is equal to 1, then taking an episode equal to 640 s (see Table 1) results in the maximum reward for one episode being 640. However, episodes are very diverse, and for some of them it is simply impossible to achieve even half of the reward due to the extreme network conditions. Figure 6 shows the cumulative rewards collected during the three validation cases, which makes it clear that MER from training is mostly influenced by the scenario's complexity and the network's unpredictability.

ABR and DRL solutions were tested on the three validation scenarios and the average outcomes were compared, see Fig. 7. DRL outperforms ABR in terms of RTT and PLR in each of the three scenarios by sacrificing some bitrate. However, for the “Moderate” and “Hard” scenarios DRL lowers GPT not so much in percentage as improves the other two parameters. It is also worth saying that an improvement of average RTT from, e.g., 80 to 60 ms is more valuable than the proportional decrease in GPT from 5 to 4 mbit/s since the RCU is controlled more reliably.

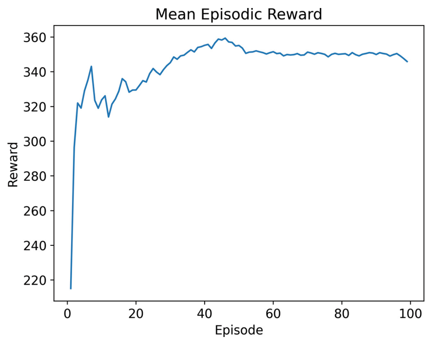


Fig. 5. Mean episodic reward during the training.

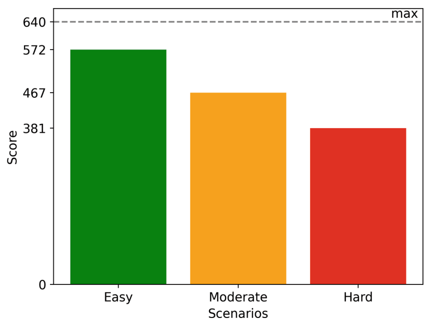


Fig. 6. Rewards for each validation scenario.

The only problem occurs with the “Easy” scenario: the agent sacrifices here too much bitrate without a need. It could be explained that the environment-agnostic agent learns more about how to find trade-offs in problematic cases:

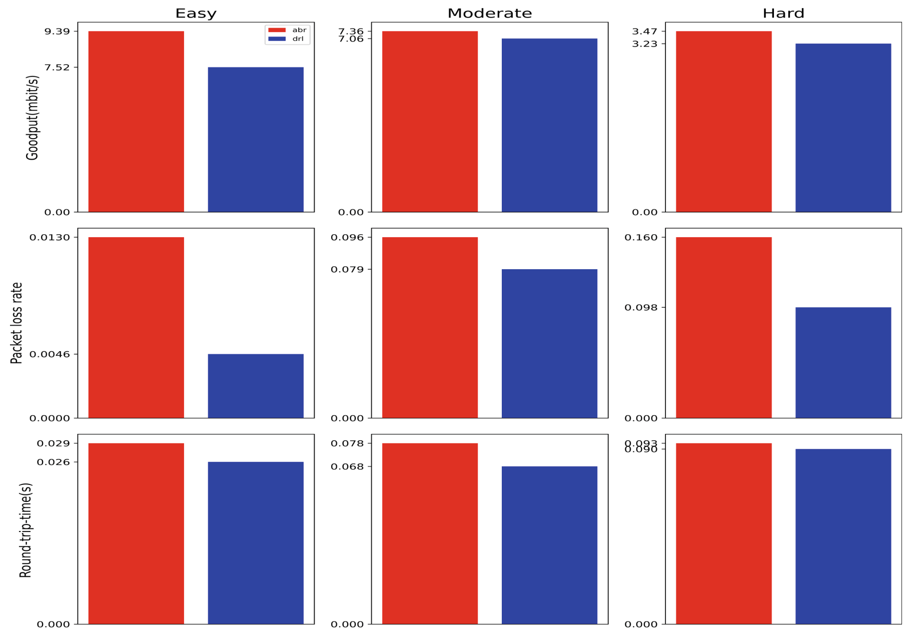


Fig. 7. The results of comparison between ABR (red) and DRL (blue) solutions on "easy", "moderate" and "hard" scenarios. (Color figure online)

if there is no loss and no strong congestion, then two of three optimization parameters (RTT, PLR) are already almost at minimum, the algorithm just needs to concentrate on keeping the transmission at the maximum rate. Such scenarios happened during the training, but they are rare because if the agent is trained on "Easy" scenarios too much, it starts losing its adaptive knowledge too fast and always selects the highest bitrate. As the result, the agent has some sort of a "fear" in "Easy" scenarios: it trembles between the two highest bitrates expecting the packet loss that may come if it stays at the maximum rate too long which will result in penalties. It is concluded that it requires adding some additional expert knowledge in its state to both perceive the adaptivity and to behave at the most optimal in all cases. This knowledge could be learned via behavioral cloning on the expert dataset.

6 Summary

The main goal of this paper was to apply the environment-agnostic and model-free DRL agent in an attempt to learn the real-data transmission problem in 5G networks and to perform better than rule-based ABR solutions without imitating their behavior. As an example problem for OC technology, it is the first step towards a data management system that could adapt to every possible network condition and control the data flow between an RCU and a human operator most optimally. The results demonstrate the ability of the DRL agent to find in most cases an effective balance between lowering the latency and loss rate without lowering the amount of transferred data too much, therefore making a remote control more reliable. While the DRL model was trained and tested only in a simulation, its design enables easy deployment on real devices in the assistance mode: a human operator might accept or ignore the suggested action at every time step.

Apart from positive results, the limits of the pure environmental-agnostic DRL approach were also clearly indicated. To be able to solve the problem in a more general way, the agent's state and reward function need to be enriched with some environmental knowledge but only to some point to avoid overfitting. That constitutes a possible future work, namely: a) to add some form of (imitated) ABR rule-based strategy to the reward function deviating from the "model-free" property, b) to add 5G channel indicators to the state that increases the quality of predictions deviating from the "environment-agnostic" property, and c) to transfer the task to a continuous action space to extend the diversity of possible actions for more optimal strategies.

Acknowledgment. This research has been partly funded by the German Federal Ministry for Digital Affairs and Transport (Bundesministerium für Digitales und Verkehr) within the project "CAPTN Förde 5G", funding guideline: "5G Umsetzungsförderung im Rahmen des 5G-Innovationsprogramms", funding code: 45FGU139_H. The authors acknowledge the financial support of the BMDV.

References

1. Biernacki, A.: Improving streaming video with deep learning-based network throughput prediction. Appl. Sci. **12**(20), 10274 (2022). https://doi.org/10.3390/app122010274
2. Cui, L., Su, D., Yang, S., Wang, Z., Ming, Z.: TCLiVi: transmission control in live video streaming based on deep reinforcement learning. IEEE Trans. Multimedia **23**, 651–663 (2021). https://doi.org/10.1109/TMM.2020.2985631
3. Dao, N.N., Tran, A.T., Tu, N.H., Thanh, T.T., Bao, V.N.Q., Cho, S.: A contemporary survey on live video streaming from a computation-driven perspective. ACM Comput. Surv. **54**(10), 1–38 (2022). https://doi.org/10.1145/3519552
4. Feng, Y., Liu, S., Zhu, Y.: Real-time spatio-temporal lidar point cloud compression (2020)
5. Huang, S., Dossa, R.F.J., Raffin, A., Kanervisto, A., Wang, W.: The 37 implementation details of proximal policy optimization (2022). https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. Accessed 08 Aug 2023
6. Jiang, J., Sekar, V., Zhang, H.: Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 97–108 (2012). https://doi.org/10.1145/2413176.2413189
7. Kaur, A., Singh, S.: A survey of streaming protocols for video transmission. In: Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence, pp. 186–191. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3484824.3484892

8. Mao, H., Chen, S., Dimmery, D., Singh, S., Blaisdell, D., Tian, Y., et al.: Real-world video adaptation with reinforcement learning (2020)
9. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The macroscopic behavior of the TCP congestion avoidance algorithm. SIGCOMM Comput. Commun. Rev. **27**(3), 67–82 (1997). https://doi.org/10.1145/263932.264023
10. Müller-Schloer, C., Tomforde, S.: Organic Computing - Technical Systems for Survival in the Real World. Birkhäuser (2017)
11. Nardini, G., Sabella, D., Stea, G., Thakkar, P., Virdis, A.: Simu5G-An OMNeT++ library for end-to-end performance evaluation of 5G networks. IEEE Access **8**, 181176–181191 (2020). https://doi.org/10.1109/ACCESS.2020.3028550
12. Nguyen, M., Lorenzi, D., Tashtarian, F., Hellwagner, H., Timmerer, C.: DoFP+: an HTTP/3-based adaptive bitrate approach using retransmission techniques. IEEE Access **10**, 109565–109579 (2022). https://doi.org/10.1109/ACCESS.2022.3214827
13. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**(268), 1–8 (2021)
14. Rekkas, V.P., Sotiroudis, S., Sarigiannidis, P., Wan, S., Karagiannidis, G.K., Goudos, S.K.: Machine learning in beyond 5G/6G networks - state-of-the-art and future trends. Electronics **10**(22), 2786 (2021). https://doi.org/10.3390/electronics10222786
15. del Río Ponce, A., Serrano Romero, J., Jimenez Bermejo, D., Contreras, L., Alvarez, F.: A deep reinforcement learning quality optimization framework for multimedia streaming over 5G networks. Appl. Sci. **12**, 10343 (2022). https://doi.org/10.3390/app122010343
16. Santos, G.L., Endo, P.T., Sadok, D., Kelner, J.: When 5G meets deep learning: a systematic review. Algorithms **13**(9), 208 (2020). https://doi.org/10.3390/a13090208
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017)
18. Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P.A., et al.: Emerging MPEG standards for point cloud compression. IEEE J. Emerg. Sel. Top. Circ. Syst. **9**(1), 133–148 (2019). https://doi.org/10.1109/JETCAS.2018.2885981
19. Smirnov, N., Tomforde, S.: Navigation support for an autonomous ferry using deep reinforcement learning in simulated maritime environments. In: 2022 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA), pp. 142–149 (2022). https://doi.org/10.1109/CogSIMA54611.2022.9830689
20. Spiteri, K., Urgaonkar, R., Sitaraman, R.K.: BOLA: near-optimal bitrate adaptation for online videos. IEEE/ACM Trans. Networking **28**(4), 1698–1711 (2020). https://doi.org/10.1109/TNET.2020.2996964