Removing human players from the loop: AI-assisted assessment of Gaming QoE

German Sviridov*†, Cedric Beliard†, Andrea Bianco*, Paolo Giaccone*, Dario Rossi†
*Politecnico di Torino, Italy – first.last@polito.it
†Huawei Technologies, Co. Ltd., France – first.last@huawei.com

***Abstract*—Quality of Experience (QoE) assessment for video games is known for being a heavy-weight process, typically requiring the active involvement of several human players and bringing limited transferability across games. Clearly, to some extent, QoE is correlated with the achieved in-game score, as player frustration will arise whenever realized performance is far from what is expected due to conditions beyond player control such as network congestion in the increasingly prevalent case of networked games. To disrupt the status quo, we propose to remove human players from the loop and instead exploit Deep Reinforcement Learning (DRL) agents to play games under varying network conditions. We apply our framework to a set of Atari games with different types of interaction, showing that the score degradation observed with DRL agents can be exploited in networking devices (e.g., by prioritizing scheduling decisions), reinforcing fairness across games, and thus enhancing the overall quality of gaming experience.**

I. Introduction

The video game industry is skyrocketing, with a market value that has surpassed that of the film industry and is expected to exceed 230 B$ by 2022 [1]. Due to its booming users base and the increasing social factor in recent games, the video game industry is starting to have a significant influence on society. Single or two-player mode games are long surpassed: most games today either offer an *optional* online mode, or can be *exclusively* played online with other users. Due to the necessity of interconnecting different players and guaranteeing high levels of gaming experience, the gaming industry has begun to attract significant interest of network operators. ISPs (such as Comcast [2]) have started to offer game-specific broadband plans that promise higher Quality of Experience (QoE) for online games. Equipment vendors (such as Ciena [3] and Barefoot [4]) put the game use-case at the heart of their network architecture evolution. Game service providers (such as Google Stadia [5]) have started to offer cloud-rendered games directly streamed to the user's home. However, initial customer experience appears to be disappointing [6] at best. This calls for further research to ensure that networks appropriately handles gaming traffic.

The large spectrum of video game genres, and the large overall video game catalog, makes the above task very complex [7]. Some games (e.g., first-person shooters or brawlers) are extremely fast paced and require swift reaction times from the players, while others (e.g., strategy or turn based games) are intrinsically less sensitive to latency. Network conditions clearly play a significant role in the user experience for the former. Large network latency or packet drops can reduce the player's performance to well below the natural score. Furthermore, among similar interactive games, the effect of a given amount of latency (or packet drop rate) may have different impact, significantly hampering playing ability in one game while and being unnoticeable to the player in another.

Due to the presence of complex in-game dynamics, it is usually wrong to assume that two apparently similar video games will lead to similar QoE under the same network conditions. Even different modes of the same first-person shooter game may lead to different responses from users depending on network conditions, as shown in [8]. This heterogeneity precludes any kind of generalization of existing results to different video games and therefore forces us to analyze QoE on a game-by-game basis. This analysis is typically done with the participation of human players and is notably a very time-consuming and expensive task that cannot keep up with the steady introduction of new games to the market.

In this paper, we take a completely different approach and advocate that it is possible to *remove human players from the QoE assessment loop*. While this may seem a bold statement at first, we base our claim on the observation that player satisfaction is naturally related to the score they are able to achieve. Whereas everybody is aware of the famous Pierre de Coubertin quote "*The important thing in the Olympic Games is not to win, but to take part*", there is little doubt that the winner of the 2019 Fortnite game competitions that brought home $ 3M [9] would think the same. We can also understand the feelings of any player who fails to win the prize or is not able to compete fairly because of bad network conditions.

Our key idea is to use scores as a proxy for user satisfaction, allowing us to automate and scale up the game assessment process. More explicitly, we propose to exploiting recent advances in the field of Artificial Intelligence (AI) (Sec. II), whereby AI agents are able to autonomously learn complex tasks such as video game playing [10] without any human intervention. Based on these agents we thus propose and implement a framework for automated assessment of game scores, and in particular of *game score degradation in presence of network impairments* (Sec. III). We use this framework on three games, gathering insights on their score degradation characteristics. These data are then used to explore how network device packet handling mechanisms might be designed to reinstate game score fairness, using scheduling as a proof of concept (Sec. IV). Finally, we discuss the issues raised in this work (Sec. V) and draw conclusions (Sec. VI).

II. Related work

A first class of work related to ours concerns QoE assessment in video games, which has been widely studied throughout the first decade of 2000s. Major effort has been put into analyzing the impact of network conditions on different games. (Sec. II-A). A second, and so far separate, class of work concerns the use of AI for video game playing. This is a much more recent field with significant contributions during the last few years (Sec. II-B).

A. Assessing video game QoE

On the track of perceived QoE, the authors of [11] analyze how the Mean Opinion Score (MOS) for the game "Call of Duty" depends on network conditions. They show that the MOS of novice players remains constant even when latency exceeds 100 ms, while it sharply decreases for experienced players after only 50 ms of latency. More quantitative results are provided in [8], which evaluates the impact of network conditions on user score in the "Unreal Tournament 2003" game. The authors show that, depending on the game mode, a score degradation of 10 %-25 % is experienced with 250 ms of latency. This is heavily in contrast with numerous previous studies which fixed the usability threshold for fast-paced games at 100 ms [7]. Given the above inconsistencies arising from human players, the research community has used in-game bots to estimate the impact of latency on the average score in "Quake" game [12]. Nevertheless, in-game bots are practically encoding expert-knowledge concerning a specific game as a set of human-programmed software heuristics.

The above approaches are thus both time consuming and cannot be generalized to other games. We seek to avoid these disadvantages by leveraging self-learning agents.

B. AI for video game playing

More recently, significant research effort has been devoted by the AI research community to develop artificial agents capable of generalizing to multiple environments. Whereas it is out of the scope of this work to provide a full coverage, for which we refer the reader to a comprehensible survey [13], we cover here the techniques we use in this work.

In particular, [10] proposes Deep Q-Networks (DQN), an application of the classic Q-Learning algorithm in the context of Deep Neural Networks. The proposed algorithm is able to learn to play Atari games by observing raw pixels and is able to achieve super-human score levels for most of them. Many improvements over the original DQN proposal have then since been proposed in follow-up research [14], [15], [16], [17] with some of them targeting more complex video games such as "Doom" game [15] or "Starcraft" game [17]. Yet in the Atari environment most of the DQN variations perform similarly to the original algorithm.

Our paper differs from this prior work in that we leverage the proposed techniques to realize the different goal of estimating QoE under network impairments.

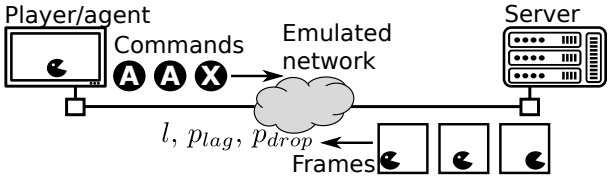


Fig. 1: Cloud Gaming (CG) emulated system.

III. Methodology

In a nutshell, our methodology consists in (i) emulating a controlled Cloud Gaming (CG) environment and (ii) training AI agents to let them play in our environment. We tune the network conditions and measure the agents' gaming performance.

A. System model of Cloud Gaming (CG)

Although being fairly novel, CG [18] is rapidly gaining momentum in the gaming community with numerous services having been released in recent years including the well-known Stadia platform [5]. Fig. 1 depicts a basic CG model.

The fundamental idea of CG consists in employing *remote rendering* of the video game environment and streaming the high-definition rendering to the clients in the form of real-time video. This is in contrast with traditional video games that are rendered on a local machine owned by the player. In the case of traditional single-player games, the entire state of the game is kept locally on the local machine, so that the perceived QoE is influenced only by the performance of the user hardware. In the case of traditional multi-player games, only metadata related to player actions are transmitted to the server, that maintains the entire state of the game.

CG on the other hand delegates the rendering responsibilities to the cloud and clients are no longer required to possess expensive dedicated hardware for local rendering. Instead, it suffices for them to be equipped with a device capable of displaying a basic video stream, such as a smartphone or a low-end laptop, for instance. CG clients interact by sending actions (e.g., keystrokes, pad or mouse movements) to the cloud, that affect the game state. The disadvantage is that, unlike in traditional video gaming rendered on local hardware, the streaming of the cloud-based rendering can be affected by varying network conditions.

From the point of view of the network requirements, CG gives rise to additional challenges as it combines elements of bandwidth-hungry applications (such as high-definition streaming) with highly delay-sensitive applications (such as real-time communication and control). It is reasonable to assume that different video games running in the cloud have roughly the same requirements in terms of bandwidth, as all require a steady stream of 4K video data. However, when it comes to latency, under same network conditions user perceived QoE may vary significantly depending on the nature of the considered video game.

In our experimental setup, both the agents and the server are in the same high-performance network, so that we are able

to measure the gaming experience (i.e., AI agent score) in both ideal conditions (sub-millisecond delay, high-bandwidth, no loss) as well as under controlled network degradation. In particular, as depicted in Fig. 1, we can actively control latency and losses at frame level, with several scenario settings and on several games, as we detailed later in the experimental section (Sec. IV).

B. Automatizing game playing

In [12] a first attempt has been made to substitute real human players by in-game bots to assess video game QoE. However, bots are not representative of real human behavior as they can exploit hidden game states, and are not limited to using only the visual information available to humans. Additionally, this approach cannot be applied when internal game states are not available, as in the case of online video games and for CG.

Recent advances in the field of AI have led to multiple models of artificial agents having great flexibility in adapting to different environments. As an example, an AI agent developed to play Atari games is equally capable of learning to play almost *any* Atari game without any human intervention or modification to the actual algorithm.

Without loss of generality, in this work we consider three different Atari games, whose screenshot is depicted in Fig. 2. The choice of using Atari games instead of more complex ones is due to the (relative) simplicity of training artificial agents. Furthermore, the Atari suite provides a large variety of games, which allow us to highlight the aforementioned heterogeneity in latency sensitiveness. We discuss how our technique applies to more complex games in Sec. V.

Analogously to real humans, AI agents are built on top of their ability to directly employ the *raw pixels as input to their decision making process*. Furthermore, as in the case of real humans, the agent decision process is built using a process of "trial and error", to progressively improve gaming performance (i.e., the game score). This allows agents to generalize to different scenarios by decoupling them from the actual game engine and makes them a promising technology for replacing humans in game QoE assessment.

C. Training AI players

More formally, we use Reinforcement Learning (RL) techniques to train our artificial agent. In RL, an artificial agent interacts with an environment (i.e., our emulated CG system) by means of a closed loop feedback system. At each time instant t the agent receives as input a representation of the environment (i.e., a state s_t that in our case is a video frame) and in turn reacts by performing an action a_t (i.e., a keypress). Following the action, the agent receives a new state s_{t+1} and the reward r_t (in our case, a possible change in the game score) corresponding to the state transition $s_t \xrightarrow{a} s_{t+1}$.

1) Practical limits of RL for games: An ideal agent must be able to perform actions so that at each time t, given the current state s_t, the action will lead to a maximum *future discounted reward* $G_t = \sum_{k \geq 0} \gamma^k r_{t+k}$. The discount rate $\gamma \in [0, 1]$ tunes the decision making process by making a compromise between immediate rewards (small γ) and possible future rewards (large γ).

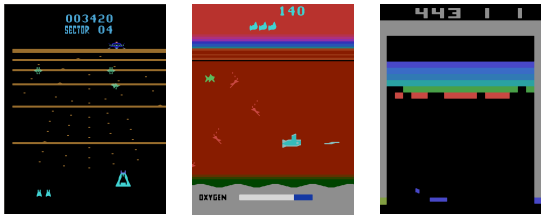


Fig. 2: The set of Atari games considered in this work. From left to right: Beamrider, Seaquest, Breakout.

As the definition of the future discounted reward depends on s_t only, it intrinsically incorporates a notion of agent behavior. Future rewards r_{t+k} are given by the current behavioral model of the agent, i.e., by its policy function $\pi(a|s)$. This function dictates the probability of taking an action a given the current state s. In principle, it is then trivial to build an explicit expression for the optimal policy $\pi^*(a|s)$ which maximizes $\mathbb{E}[G_t]$. However, explicitly computing $\pi^*(a|s)$ becomes burdensome when the number of states grows large.

2) Human-like "trial and error" and Q-Learning: A first practical limitation of RL is that learning the optimal policy proves to be challenging due to its excessive computation cost. Thus, multiple *approximate algorithms* have been proposed, including Q-Learning, which is among the most widely used and studied approximate RL techniques. The main concept behind Q-Learning is finding an optimal state-action function $Q^*(s, a)$, which rewards $\mathbb{E}[G_t]$ by performing action a in state s at time t. Q-Learning employs a fundamental result from the field of dynamic programming, namely the *Bellman equation*, that determines Q^* in an iterative fashion. Starting from sub-optimal Q and simply selecting at each time step the action a that maximizes Q, i.e., $a = \arg\max_{a'} Q(s, a')$, and updating the new estimate of Q, provides guarantees of converge with probability 1 to optimal Q^* under non-restrictive conditions. From Q^* it is immediate to define the optimal policy $\pi^*(a|s)$. *This allows agents to improve their score over time, learning to play like humans do.*

3) Human-like vision and CNN: A second practical limitation of RL is that approximate methods reach their limits when the state space becomes very large since building an explicit Q function is then unfeasible. Consider a simple arcade game whose state is represented by the current frame. Even a monochrome resolution as small as 64×64 pixels will lead to a state space of $2^{64 \times 64}$.

Deep Q-Network (DQN) [10] has been proposed as a solution to overcome the limitation of the exploding state-space, by leveraging recent advances in the field of computer vision achieved through Deep Learning (DL) techniques. In particular, instead of employing an exhaustive Q function, DQN employs a Convolutional Neural Network (CNN) to infer the Q function, starting from the raw pixels of the frames. *This approach allows agents to play any game, as a human would*

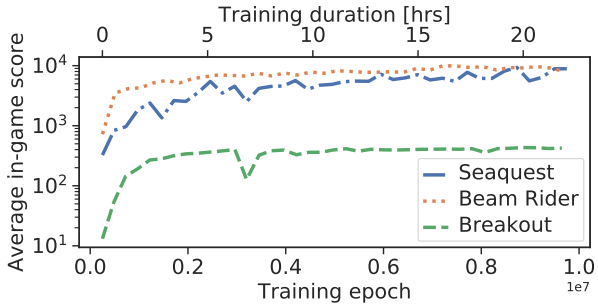


Fig. 3: Training performance of DQN for three Atari games.

do, through a CNN equivalent of a visual interaction with a rendered CG game.

4) Training in practice: We use OpenAI Gym toolkit to train agents to play three different Atari games shown in Fig. 2 using the model described in [10]. The agents are trained on two NVIDIA V100 GPUs, letting them interact with our emulated CG in unperturbed settings, i.e., with ideal network performance. We report the training results in Fig. 3 showing the average score achieved by the agent as a function of the number of update steps. On average a training time of 22 hours (equivalent to 10M training epochs) is sufficient to reach the maximum score for all of the considered games.

IV. Experimental evaluation

In this section, we first provide details about the experimental setup (Sec. IV-A). We then use the agents trained as described in the previous section to assess the impact of network conditions on score degradation (Sec. IV-B). Finally, we show how we can exploit our findings to improve the QoE by acting on network scheduling (Sec. IV-C).

A. Network model

With reference to the CG architecture depicted in Fig. 1, we model the communication channel between the player and the game server as an asymmetric link characterized by the parameters depicted in Table I. Notably, we tune (i) the probability that a frame is delayed p_{lag}, (ii) the amount of latency l and (iii) the keystroke loss probability p_{drop}. Note that l is defined in respect to the action taken during the delayed frame and is usually referred to as *lag*.

We consider the case where the communication channel is adequately over-provisioned in terms of bandwidth to support multiple simultaneous games, thus no network congestion is experienced. For the sake of simplicity, we assume no inter-frame video encoding. Thus for each update of the game state a full frame containing the new state of the visual playing environment is sent to the client. Frames are sent at a constant rate of 60 frames per second. This simplistic setting is a reasonable simplification given that, for the time being, we do not desire to tackle the problems of bandwidth adaptation logic in the video streaming portion of the CG (see Sec. V).

Frame delays and losses can thus be synthetically added to control network conditions. For the sake of simplicity, we consider simple independent probabilistic models to add delay or to lose a packet. With probability p_{lag} we add a given amount of lag l for each frame transmitted from the server. At the client side, at time t_0, if the received frame relates to a time instant $t > t_0$, then the frame is rendered, otherwise the received frame is discarded and the last rendered frame is re-rendered for that time instant, essentially simulating a game freezing behavior. Note that for $p_{lag} = 1$ every frame is delayed by l corresponding to the case of a constant lag scenario. However, emulating correlated delays and losses (as would happen in the case of transient bandwidth bottlenecks) can be easily realized by using simple Markovian models. *The delay in the reception of the new frames will alter the agent's ability to play, exactly as would happen to humans in the case of sluggish network conditions.*

One last point is worth elucidating: the agent sends back an action for every received frame in the form of a keystroke. For the case of $p_{drop} > 0$, whenever a keystroke is dropped, the game server interprets it as a *no action*. The environment is then advanced by one frame without executing any action.

TABLE I: Channel parameters employed in the experiments

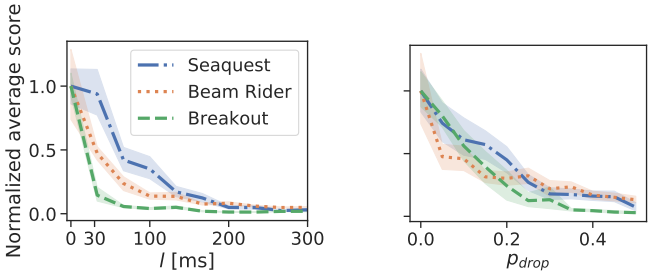
Variable	Description	Range
p_{drop}	Per-keystroke drop probability	[0, 0.5]
p_{lag}	Per-frame lag probability	[0, 0.8]
l	Lag duration	[0, 300] ms

B. Assessing score degradation

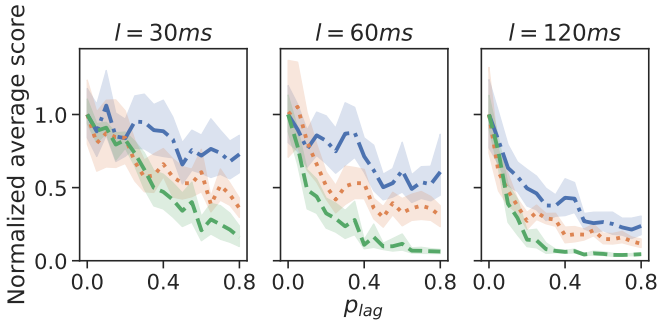
We emulate a game played over synthetic network conditions using the trained agents. As for traditional QoE assessment, our goal is to observe how our agent playing capabilities are affected by network conditions. As typically done with human players [8], we quantify the influence of the network on QoE by considering the average score that the agent is able to achieve in a perturbed scenario. To compare scores across games, we normalize the score over the average score achieved in a non-perturbed scenario: a normalized score ≈ 1 corresponds to no noticeable game impairment (i.e., highest MOS in an Absolute Category Rating scale), whereas a score ≈ 0 corresponds to the most severe degradation (i.e., lowest MOS). We perform a set of experiments with a non-ideal communication channel and report our findings in Fig. 4. Furthermore, we report the 95% confidence interval of the obtained score to show the significance of the results.

1) Fixed latency: We consider the scenario of a communication channel with an added network latency. We vary this lag between 0 and 300 ms and observe the score achieved by the agent. Fig. 4a shows that all three games have a sharp decrease in the score when additional latency is added. However, for low lags there is a large discrepancy in the score degradation across games: Breakout, Beamrider and Seaquest exhibit a significant, moderate and negligible score degradation, respectively.

This provides initial insights on the dynamics of three games. Breakout dynamics are very fast paced, leaving small

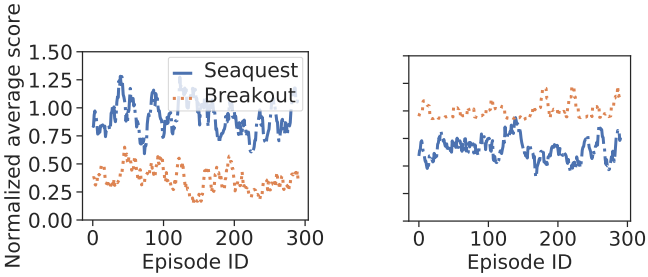


(a) Deterministic frame lag. (b) Keystroke loss probability.



(c) Probabilistic frame lag.

Fig. 4: Score degradation under perturbed network conditions



(a) Deterministic frame lag. (b) Keystroke loss probability.

Fig. 5: Normalized game scores achieved by AI agents under Blind (left) vs QoE-aware (right) schedulers

error margins. On the contrary, in Seaquest and Beamrider the game dynamics are more forgiving, with the agent being able to easily avoid obstacles and shoot at enemies even in the case of a delayed scenario. *This allows us to conclude that, in the low lag regime, not all games are equally penalized by the same network conditions.*

Second, it is also clear that after 100 ms of extra latency, the latter two games become unplayable. This is not different from the human perception timescale, where 100 ms is typically considered as a threshold that significantly affects the ability to retain control in interactive tasks or communication [7]. This is also found in cloud-based games, which strive to maintain latency at an even lower level [19]. *Games are therefore unplayable in these conditions and AI agents confirm this expectation.*

2) Random keystroke drop: We repeated the experiment by this time varying the keystroke drop probability p_{drop}. Fig. 4b summarizes our findings: all three games have a similar score degradation as a function of p_{drop} with Seaquest being the most tolerant among the three. *This allows us to deduce that latency is more important than drops, as dropped actions are naturally repeated by players, realizing a sort of implicit Forward Error Correction (FEC) mechanism.*

3) Random latency: We perform further evaluations by considering the case of probabilistic latency. Results depicted in Fig. 4c show the score evolution as a function of both l and p_{lag}. These results closely mimic those obtained in Fig. 4a with similar score degradation for all three games. As in the case of fixed latency, Seaquest is able to tolerate higher latency, while Breakout has rapid score degradation even for small l and p_{lag}.

The important takeaway from the figures is that our method is automated and is able to gather very fine-grained score degradation maps for a combination of parameters. The bottleneck is mostly represented by agent training time (i.e., 22 hours per agent in our scenarios), whereas the experiments for Fig. 4 took less than 30 minutes each.

C. Achieving per-game QoE fairness

Knowing a fine-grained QoE response to network conditions as provided by our methodology opens new opportunities for in-network QoE management such as resource placement, or QoE-aware packet scheduling.

Building upon the results of Fig. 4a, we make a simple proof of concept, where we take scheduling decisions that are aware of the heterogeneous sensitivity of games to network impairment. ISPs and CG providers need to perform resources arbitration in the case of multiple concurrent players. The objective can be formulated loosely as enforcing the inter-game fairness: in the case of two or more concurrent game sessions, a QoE-aware scheduler could prioritize the most latency-sensitive games, as the experience of players of the other games would be less affected by extra latency.

We thus perform an experiment in which we alter the scheduling policy for two concurrent sessions of the Breakout and Seaquest games. Fig. 5 shows the time evolution of the score of both games. Both sessions share the same bottleneck link which adds a per frame delay of 30 ms with probability $p_{lag} = 0.5$, and both games are equally treated by a Round Robin (RR) scheduler. Fig. 5a shows that although both game flows observe the same latency a big discrepancy is experienced in the obtained scores. Breakout is heavily affected by the network impairments, achieving an average normalized score of 0.35, whereas Seaquest is almost unperturbed, achieving 0.96 average normalized score.

Using a game QoE-aware scheduler allows us to perform scheduling based on the expected impact on gaming performance. As a proof of concept, we switch the scheduling policy to Strict Priority (SP) and assign the highest priority to Breakout flows. In this scenario, as reported in Fig. 5b, the Breakout agent observes an ideal channel and is able to achieve a

normalized score of 1 while Seaquest sees its normalized score decrease to 0.65. However, this priority scheduling increases the average game score which goes from 0.67 in the case of RR scheduling, to 0.82 in the latter case. The considered scenario is, of course, only considered as a proof of concept to show the feasibility of the proposed approach: in practice one would use advanced weighted scheduling mechanisms, with weights deduced from results in Fig. 4. The evaluation of this mechanism is left for future work.

V. Discussion

Our proposal raises some interesting questions that this paper was not able to solve and which we now discuss briefly.

A. Game stages classification

During our analysis, we considered only games requiring the same playing style throughout the entire game. Modern games are typically composed of a series of different stages, e.g., action stages, exploration stages, dialogue stages, etc. Identifying different stages would lead to an even finer grained control over the resources to be allocated to single games. As previously shown, low lags added in a fast-paced action stage may lead to catastrophic performance degradation, whereas a 500 ms lag in a dialogue stage would be hardly noticed by the player. Automatic game stages detection and classification can be achieved using a methodology similar to the one we presented by observing how the agent reacts to channel perturbation throughout its play-through. *Including more diversity and newer games is part of our ongoing work.*

B. Delay-tolerant agents

Our results show that latency plays a fundamental role for the performance of the agent. These results are consistent with other studies performed with real human players [8], [11]. On the other hand, studies such as [20] showed that, to a certain extent, experienced players are able to tolerate the effects of latency on their gaming performance, and incur smaller score penalties. In the field of DRL, there has been limited interest in building agents resilient to delayed environment response. However, there exists a significant body of work in the field of classical RL that considers these issues [21] and provides strong theoretical results on model requirements [22]. *We are currently experimenting with delay-tolerant DRL agents to refine our framework.*

C. Agent calibration

Building artificial agents capable of adapting to scenarios with network impairments introduces another degree of freedom in the assessment of QoE by tuning the "experience level" [20] or the "play style" [23] of the artificial player. In [23] significant steps have been made towards creating agents with different play styles, that can make AI performance closer to human performance. *Validating the results that we gathered in this paper with a study including real human subjects is a necessary step in our future work agenda.*

VI. Conclusion

In this paper we propose a novel methodological approach for efficient and reliable QoE assessment in Cloud Gaming scenarios. At its core, the proposed approach employs artificial players instead of real humans to assess game session QoE under different network impairments. We show performance results for three Atari games for different communication conditions that are in line with related literature employing human subjects.

We argue that this development yields a versatile and efficient tool for automating QoE assessment, by employing artificial players instead of real human subjects. We expect this to bring a substantial reduction in the cost and complexity of the entire process, while introducing a rigorous, methodological and scalable strategy for the assessment of game QoE.

References

[1] https://www.pcgamesn.com/game-industry-revenue-2018.
[2] https://www.xfinity.com/esports.
[3] https://www.ciena.com/.
[4] https://canopusnet.com.
[5] https://stadia.google.com.
[6] https://www.forbes.com/sites/paultassi/2019/11/18/google-stadia-launch-review-a-technical-conceptual-disaster/.
[7] M. Claypool and K. Claypool, "Latency can kill: precision and deadline in online games," in *ACM Multimedia Systems Conference*, 2010.
[8] T. Beigbeder, R. Coughlan *et al.*, "The effects of loss and latency on user performance in unreal tournament 2003®," in *ACM SIGCOMM workshop on Network and system support for games*, 2004.
[9] https://www.cnbc.com/2019/07/29/fortnite-world-cup-us-teen-wins-3-million-at-video-game-tournament.html.
[10] V. Mnih, K. Kavukcuoglu *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
[11] R. Amin, F. Jackson *et al.*, "Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2," in *Int. Conf. on Human-Comp. Inter. (HCI)*, 2013.
[12] S. Zander, I. Leeder *et al.*, "Achieving fairness in multiplayer network games through automated latency balancing," in *International Conference on Advances in computer entertainment technology*. ACM, 2005.
[13] N. Justesen, P. Bontrager *et al.*, "Deep learning for video game playing," *IEEE Transactions on Games*, 2019.
[14] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *AAAI Fall Symposium Series*, 2015.
[15] G. Lample and D. S. Chaplot, "Playing FPS Games with Deep Reinforcement Learning," in *Conf. on Artificial Intelligence (AAAI)*, 2017.
[16] M. Wydmuch, M. Kempka *et al.*, "Vizdoom competitions: playing doom from pixels," *IEEE Transactions on Games*, 2018.
[17] V. Zambaldi, D. Raposo *et al.*, "Relational deep reinforcement learning," *arXiv:1806.01830*, 2018.
[18] W. Cai, R. Shea *et al.*, "A Survey on Cloud Gaming: Future of Computer Games," *IEEE Access*, 2016.
[19] M. Claypool and D. Finkel, "The Effects of Latency on Player Performance in Cloud-based Games," in *IEEE/ACM Workshop on Network and System Support for Games (NetGames)*, 2014.
[20] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *International workshop on Network and operating systems support for digital audio and video*. ACM, 2002.
[21] T. J. Walsh, A. Nouri *et al.*, "Learning and planning in environments with delayed feedback," *Autonomous Agents and Multi-Agent Systems*, 2009.
[22] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE Transactions on Automatic Control*, 2003.
[23] M. Jaderberg, W. M. Czarnecki *et al.*, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *arXiv:1807.01281*, 2018.