Reinforcement Learning for Game Personalization on Edge Devices

Anand Bodas, Bhargav Upadhyay, Chetan Nadiger, Sherine Abdelhak
Intel Corporation
e-mail: {anand.v.bodas, bhargav.s.upadhyay, chetan.hemanth.nadiger, sherine.abdelhak}@intel.com

***Abstract*—Good progress has been shown recently in the area of active learning, specifically, Reinforcement learning (RL). In this paper, the authors show how RL can be used to personalize games based on user-interaction with the game. The work uses Deep Q network models (DQN) and the open source framework OpenAI to build an RL model that is able to optimize the gamer's engagement level in a game. The authors define an example quantitative measure of gamer engagement and incorporate that into the DQN learning reward function. The gamer experience optimization is empirically demonstrated using a game of Pong. Simulation testing and analysis of results indicate adapted RL models increase engagement reward values, thus enhancing gamer experience. The contribution of this paper is twofold: (1) using RL, it paves the path for wider adaptation to user-behavior, starting with gaming, and (2) it shows analysis and feasibility of an RL algorithm on an edge device (Personal Computer) in real-time.**

Keywords-reinforcement learning; computer games; artificial intelligence; game personalization, edge computing

I. INTRODUCTION

Computer games typically involve human(s) who play against a Non-Player Character(s) (NPC). Among other factors, the NPC behavior is in reaction to the human actions and state of the game. Since 1950s, the development of NPC programs has been a major focus of research for the domain of artificial intelligence (AI) [1].

Significant progress has been made in applying AI technique like Reinforcement learning (RL) to game play. Recently, RL has been very successful in building AI players for simple Atari games [2] as well as for more sophisticated games like Go [3]. These techniques build AI agents to play a particular game / game type. The agent's objective is to play against the human opponent and be able to win the game.

This paper focusses on 'personalization' of NPCs to specific human players. Personalization can be driven by player's engagement and/or emotions; and imply influencing these factors directly and/or indirectly. It is worth noting that the techniques proposed in this paper can be expanded beyond gaming to other domains like media consumption, and beyond NPC, to game scenery, music, etc.

The authors chose a 2-player game of Pong to prove the personalization functionality, as well as its feasibility on edge device (Personal computer). The work presented in this paper describes training an RL model that can play the game of Pong against a human player with objective of increasing gamer engagement. The authors propose an example quantitative measure of gamer engagement specific to the Pong game (Refer to section IV). Though this paper presents the concept for a specific game (Pong), the proposed technique can be extended to other similar games. Please note in this paper, the terms NPC and RL agent mean the same and are used interchangeably.

The paper is organized as follows. Section II reviews the literature around RL in gaming and describes the unique contribution of this paper. Section III gives a background of the proposed approach including a brief overview of Deep Q Network architecture, the Pong game and the frameworks used. Section IV describes the gamer engagement reward function for the game of Pong and the personalized RL agent training. Section V presents the results and discusses the outcomes. Section VI concludes the paper with summary and planned future work.

II. BACKGROUND AND RELATED WORK

Reinforcement learning (RL) is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward [4]. A high-level depiction of that is shown in figure 1.

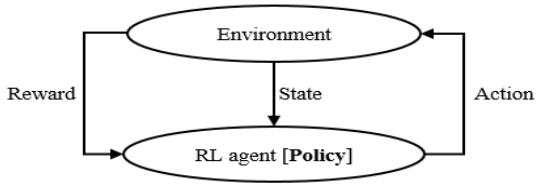

Figure 1. Reinforcement learning.

The aim of RL is to learn a 'Policy' that correctly maps the state of the environment to appropriate action to be taken by the agent. 'Appropriate' actions produce higher rewards from the environment. The environment is generally modelled as a Markov Decision Process.

Volodymyr Mnih et.al [2] have developed the Deep Q Network (DQN) model that can be trained to play Atari games. DQN combines reinforcement learning with a class of artificial neural network known as deep neural networks (DNN). Among many other contributions, the model's ability to learn any Atari game without explicitly knowing the rules of the game is crucial. This is achieved by using game screen pixels as the only input. This paper builds on that approach and modifies it to achieve a new end-goal,

978-1-5386-5384-5/18/$31.00 ©2018 IEEE

which is around personalization rather than winning a challenge.

Julian Togelius *et. al* [9] report work on game personalization by evolving the rules of the game and do not address learning a NPC strategy. Zhang, Haifeng, et al [10] too take the same approach as [9] for personalization. They evolve skill- matched mazes in the game to keep the gamer engaged.

The key contributions of this work are to empirically show how RL can be used for enhancing gamer engagement and analysis of adaptation on edge devices. This is achieved by building an intelligent RL-based NPC which can enhance the game play experience by online personalization.

III. Proposed Approach: Background

This section sets the stage for the proposed approach by going over DQN learning, the Pong game and the frameworks used in this work.

A. DQN Learning

Volodymyr Mnih et.al [2], in their seminal work on DQN use game screens as input to the DNN and produces Q values for each valid action. The target Q values and current Q values are incorporated into the loss function according to the Bellman equation. The DQN architecture maintains a separate target neural network for target Q values, which draws samples from the 'experience replay' buffer for mini-batch learning. The DQN algorithm presented in [2] works for discrete action space (as in Atari games) and has been shown to successfully master the game and sometimes even exceeding human performance.

The rewards used in the original DQN learning implementation is typically the game score. This reward function in shown in equation 1.

$$R_t = A_{t-1} - C_t \tag{1}$$

Update:

$$A_t \leftarrow R_t$$

where,

R_t: Reward value at time step t

$0 \leq A_t \leq P$: Accumulated points till time step t. P is the maximum game points that can be obtained.

$0 \leq C_t \leq 1$: Points at current time step t

The DQN learning algorithm aims to maximize reward function in equation 1 i.e. tries to win the game.

B. Pong Game

A typical pong game screen is shown in figure 2.

The human player and RL agent move the paddle up/down/left/right to hit the ball and try to make the other player miss the ball. If the opponent misses the ball, the players' score increases. The player who reaches score of 21 (P in equation 1), wins the game. This is termed as one episode of the game.

Figure 2. Pong game screen.

C. Frameworks

The authors use Open AI baselines [7] for DQN learning implementation and Roboschool [8] for multiplayer Pong environment. The original DQN training for Atari games receives Pong game screens as input, while Roboschool Pong environment gives an 11 dimensional game state vector as input. The authors trained a generic Pong playing (GPP) RL agent which receives this 11 dimensional state vector from Roboschool environment as input. The GPP RL agent can play against different opponents with the aim of winning. The GPP RL agent training and its personalization is explained in section IV.

IV. Proposed Approach : RL Agent Personalization

The authors define a quantitative scalar reward function which indicates gamer engagement levels. Among many other indicators of gamer engagement, correct skill match between NPC and the opponent is crucial for enhanced gaming experience [5]. In this paper, the authors use a composite reward function which implicitly tries to match the skill level of NPC with that of the opponent in the Pong game. This is shown in equation 2.

$$\begin{aligned} R_t &= W_1 \;\; if\ RL\ agent\ loses\ the\ point \\ R_t &= W_2 \;\; for\ every\ rally\ count\ increment \\ R_t &= W_3 \;\; if\ rally\ ends \end{aligned} \tag{2}$$

where,

R_t: Reward value at time step t

W_1, W_2, W_3 are tunable scalar parameters

Referring to equation 2, whenever any player hits the ball, a rally is said to have occurred and the rally count is incremented by one. The authors use rally count as an indicator of gamer engagement. Longer rallies make the game competitive and hence improve experience. The composite reward function balances between longer rallies and winning/losing of the RL agent by having suitable values for parameters W_1, W_2 and W_3. The values used in this work are $W_1 = -1$, $W_2 = 0.5$ and $W_3 = -0.5$. It is possible to use a different reward function for different games for gamer engagement quantification.

Building an RL agent that can optimize the composite reward function (equation 2) involves two major steps. Step 1 is to train a generic Pong playing (GPP) RL agent from 'scratch'. This GPP RL agent uses game score as its reward function (Equation 1) and learns to win the game. Step 2

uses the policy of the GPP RL agent and fine-tunes it using the gamer engagement reward function (Equation 2). Step 2 over time makes the RL agent play at similar skill level as the opponent player. Both step 1 and step 2 are explained in detail in section IV (A) and IV (B).

A. Building Generic Pong Players

The authors use the DQN implementation in OpenAI baselines framework with modified neural network architecture to suit the environment (game) state input from Roboschool framework. The input state vector to the DQN is a 13 dimensional vector consisting of positions and velocities of the paddles and the ball along with a timeout. The fully connected neural network architecture of the DQN used is as specified below.

Number of hidden layers: 2
Number of hidden neurons: 130 (1st layer) and 90 (2nd layer)
Number of input neurons: 13
Number of output neurons: 2
Activation function: Rectified Linear Unit

The DQN implementation uses a discrete action space to produce 2 actions corresponding to horizontal and vertical motion of paddle, including its direction. These actions are converted into a constant paddle velocity (depending upon the direction) and are given as input to the Roboschool Pong environment. After defining the network architecture and other learning parameters, the authors train the GPP RL agent.

Roboschool provides two sample trained neural network agents (say Roboschool NN agents) which can play Pong with continuous action values. The authors use these two neural network agents to test and rate the skill level of the trained DQN agents. The skill of a trained DQN agent is defined as number of episodes it wins against the Roboschool NN agents over a series of 100 episodes. By running the DQN learning algorithm for different number of episodes, the authors arrive at RL agents of different skill levels. Table 1 shows an example of few such RL agents.

TABLE I. DIFFERENT SKILL LEVEL RL AGENTS

RL agent number	Number of training game episodes	Skill Level
P1	150	Low
P4	500	Medium
P7	900	High

These RL agents are not generalized and over-fit to the Roboschool NN agent against which they were trained. To build a GPP RL agent, the authors train a new RL DQN model by training against multiple RL agents of varying skills via bootstrapping. This completes step 1.

B. RL on Edge Devices: Personalization

In step 2, the GPP agent, which plays to win, is pitted against opponents of different skill levels and it uses the composite reward function (equation 2) to personalize its play and increase the rally length against the opponents. For consistency of results during testing, the authors utilize different 'fixed' skilled RL agents as proxy for human opponents. Here, fixed means that the policy of RL agents is not updated further.

One challenge during personalization is that there should not be a significant drop in gamer engagement even during starting of training. This means, training an RL agent from 'scratch' even with composite reward function (equation 2) is not feasible due to the high amount of 'mistakes' the RL agent makes in the starting phase of training. Thus, instead of starting the learning from random weights, the authors pre-initialize the DQN model with weights from GPP model. This pre-initialization makes the personalization faster without drop in gamer engagement metric.

Another challenge with personalization is the time required to increase the gamer engagement. The exploration policy rate [2] is a parameter which impacts the adaptation time. Because, the model is pre-initialized with GPP weights, the authors use a low value for exploration policy rate to decrease the adaptation time. This does not affect the game experience as empirically shown in results section V. The RL model built from fine-tuning is termed as Personalized Pong Player (PPP) RL agent.

The training and personalization is done on a CPU; a 7th Gen Core i7 4-Core desktop. Tensorflow with MKL [11] support is used and the parameters (KMP_BLOCKTIME, KMP_AFFINITY, OMP_NUM_THREADS, inter_op_parallelism_threads, intra_op_parallelism_threads) are tuned to get better performance as explained in [12]. The results and analysis of experiments conducted are presented in the section V.

V. RESULTS AND ANALYSIS

As explained in section IV (A), many fixed RL agents are trained which can be used to assess the personalization. Two experiments were conducted in this regard to ascertain the degree of personalization and time required for adaptation.

A. Degree of Personalization

Figure 3 shows how the GPP and PPP RL agent performs against fixed RL agent of different skill levels in terms of average rally count.

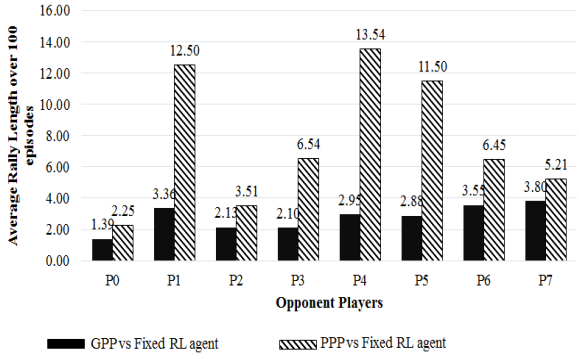

Figure 3. Comparision of average rally length by GPP and PPP.

The GPP and PPP models are made to play against fixed RL agents (P0 to P7) over an interval of 100 episodes. The P0 to P7 players are ranked in ascending order of their skill levels. The graph proves that PPP in all cases provides higher rally lengths irrespective of the skill levels, thus providing higher gamer engagement. It was visually noted that for 'weaker' opponents (P0, P1, P2) the RL agent mostly hits the ball directly towards the opponent paddle and minimizes bounces off the walls. The direct hit and bouncing off wall is depicted in figure 4 and 5.

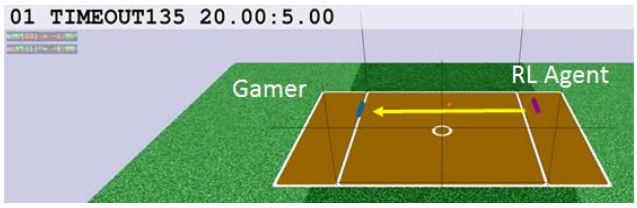

Figure 4. Depiction of direct hit by RL agent.

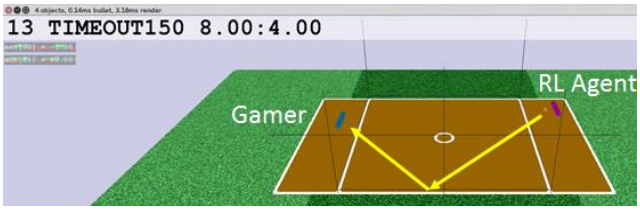

Figure 5. Depiction of bouncing off wall hit by RL agent.

It is also noted that the average rally length difference between GPP and PPP is lower against very low and very high skilled players. It is verified through visual observations that it is because low skill players miss the ball a lot and hence PPP is not able to increase the rally length. In high skilled player's case, the opponents sometimes beat the PPP and hence cause lower rally lengths.

B. Adaptation Time

As mentioned in section IV (B), during personalization on edge, the exploration policy rate is kept to a low initial value of 30%. An analysis of time taken during GPP personalization to reach a peak average rally length is shown in figure 6. This is measured in terms of number of game episodes.

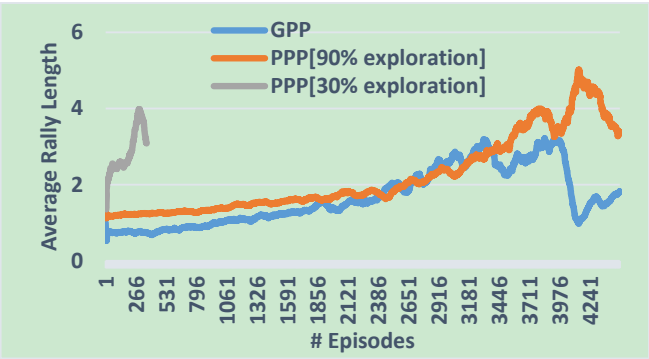

Figure 6. Comparision of time taken to reach peak average rally length.

Referring to figure 4, it can be observed the PPP RL model obtained from 30% exploration takes significantly lesser number of training episodes to reach an average rally length of 4. This average rally length is comparable to PPP obtained from 90% exploration rate which takes significantly more number of episodes. This shows faster adaptation without drop in experience.

VI. CONCLUSIONS

This paper investigates the usage of Reinforcement Learning for personalization applications and proposes a RL based method for gamer experience enhancement. The authors define a gamer engagement reward function and empirically prove how RL optimizes this reward thus achieving personalization. The paper proposes mechanisms to overcome challenges in using RL for personalization on edge devices.

As further extension of this work, there can be more sophisticated reward functions (incorporating user behavior/emotions). This could be derived from subjective experience analysis on human gamers. The personalization methods proposed in this paper can also be extended to non-gaming domains like media consumption and natural language dialogue systems.

REFERENCES

[1] Galway, Leo, Darryl Charles, and Michaela Black. "Machine learning in digital games: a survey." Artificial Intelligence Review 29.2 (2008): 123-161.

[2] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

[3] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

[4] https://en.wikipedia.org/wiki/Reinforcement_learning (Accessed on 11th Dec, 2017)

[5] Gregory, E. (2008). Understanding Video Gaming's Engagement: Flow and Its Application to Interactive Media. Media Psychology Review. Vol. 1(1)

[6] Brockman, Greg, et al. "OpenAI gym." arXiv preprint arXiv:1606.01540 (2016).

[7] Dhariwal, et al. "OpenAI Baselines", https://github.com/openai/baselines (Accessed on 11th Dec, 2017)

[8] https://github.com/openai/roboschool (Accessed on 11th Dec, 2017)

[9] Togelius, Julian, and Jurgen Schmidhuber. "An experiment in automatic game design." Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On. IEEE, 2008.

[10] Zhang, Haifeng, et al. "Learning to Design Games: Strategic Environments in Reinforcement Learning." (2017).

[11] https://software.intel.com/en-us/mkl (Accessed on 11th Dec, 2017)

[12] https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn (Accessed on 11th Dec, 2017).