PLAYING SNES IN THE RETRO LEARNING ENVIRONMENT

Nadav Bhonker*, Shai Rozenberg* and Itay Hubara
Department of Electrical Engineering
Technion, Israel Institute of Technology
(*) indicates equal contribution
{nadavbh,shairoz}@tx.technion.ac.il
itayhubara@gmail.com

ABSTRACT

Mastering a video game requires skill, tactics and strategy. While these attributes may be acquired naturally by human players, teaching them to a computer program is a far more challenging task. In recent years, extensive research was carried out in the field of reinforcement learning and numerous algorithms were introduced, aiming to learn how to perform human tasks such as playing video games. As a result, the Arcade Learning Environment (ALE) (Bellemare et al., 2013) has become a commonly used benchmark environment allowing algorithms to train on various Atari 2600 games. In many games the state-of-the-art algorithms outperform humans. In this paper we introduce a new learning environment, the Retro Learning Environment — RLE, that can run games on the Super Nintendo Entertainment System (SNES), Sega Genesis and several other gaming consoles. The environment is expandable, allowing for more video games and consoles to be easily added to the environment, while maintaining a simple unified interface. Moreover, RLE is compatible with Python and Torch. SNES games pose a significant challenge to current algorithms due to their higher level of complexity and versatility. A more extensive paper describing our work is available on arXiv[1].

1 INTRODUCTION

Creating an artificial intelligence (AI) agent whose behavior is based solely on raw high-dimensional input data such as image or sound is a difficult and important task in the field of Reinforcement Learning(RL). Recent breakthroughs in the field allow its utilization in real-world applications such as autonomous driving (Shalev-Shwartz et al., 2016), navigation (Bischoff et al., 2013) and more. Agent interaction with the real world is usually either expensive or not feasible, as the real world is far too complex for the agent to perceive. Therefore in practice the interaction is simulated by a virtual environment which receives feedback on a decision made by the algorithm. Modern games often present problems and tasks which are highly correlated with real-world problems. For example, an agent that masters a racing game, by observing only simulated driver's view screen as an input, may be useful for the development of an autonomous driver. For high-dimensional input, the leading benchmark is the Arcade Learning Environment (ALE) (Bellemare et al., 2013) which provides a common interface to dozens of Atari 2600 games, each presents a different challenge. The main challenge posed by ALE is to successfully play as many Atari 2600 games as possible (i.e., achieving a score higher than an expert human player) without providing the algorithm any game-specific information (i.e., using the same input available to a human - the game's screen and score). A key work to tackle this problem is the Deep Q-Networks algorithm (Mnih et al., 2015), which made a breakthrough in the field of Deep Reinforcement Learning by achieving human level performance on 29 out of 49 games. In this work we present a new environment — the Retro Learning Environment (RLE). RLE sets new challenges by providing a unified interface for Atari 2600 games as well as more advanced gaming consoles. As a start we focused on the Super Nintendo Entertainment System (SNES). Out of the five SNES games we tested using state-of-the-art algorithms, only one was able to outperform an expert human player.

[1] arxiv.org/abs/1611.02205

2 THE RETRO LEARNING ENVIRONMENT

2.1 SUPER NINTENDO ENTERTAINMENT SYSTEM

The Super Nintendo Entertainment System (SNES) is a home video game console developed by Nintendo and released in 1990, supporting a total of 783 games. Table (1) presents a comparison between Atari 2600, Sega Genesis and SNES game consoles, from which it is clear that SNES and Genesis games are far more complex.

2.2 IMPLEMENTATION

To allow easier integration with current platforms and algorithms, we based our environment on the ALE, with the aim of maintaining as much of its interface as possible. While the ALE is highly coupled with the Atari emulator, Stella[2], RLE takes a different approach and separates the learning environment from the emulator. This was achieved by incorporating an interface named LibRetro (libRetro site), that allows communication between front-end programs to game-console emulators. Currently, LibRetro supports over 15 game consoles, each containing hundreds of games, at an estimated total of over 7,000 games that can be supported using this interface. We chose to focus on the SNES game console implemented using the SNES9X[3] as it's games present interesting, yet plausible to overcome challenges. Additionally, we utilized the Genesis-Plus-GX[4] emulator, which supports several Sega consoles: Genesis/Mega Drive, Master System, Game Gear and SG-1000.

2.3 RLE'S INTERFACE

RLE provides a unified interface to all games in its supported consoles, acting as an RL-wrapper to the LibRetro interface. Actions have a bit-wise representation where each controller button is represented by a one-hot vector. Therefore a combination of several buttons is possible using the bit-wise OR operator. The environments observation is the game screen, provided as a 3D array of 32-bit pixels with dimensions that vary depending on the game. The reward can be defined differently per game, usually we set it to be the score difference between two consecutive frames. By setting different configuration to the environment, it is possible to alter in-game properties such as difficulty (i.e easy, medium, hard), its characters, levels, etc. RLE is fully available as open source software under GNU's General Public License[5]. The environment is implemented in C++ with an interface to algorithms in C++, Python and Lua. Adding a new game to the environment is a relatively simple process. Additionally, the environment supports running multiple agents simultaneously against each other on certain games.

Table 1: Atari 2600, SNES and Genesis comparison

	Atari 2600	**SNES**	**Genesis**
Number of Games	565	783	928
CPU speed	1.19MHz	3.58MHz	7.6 MHz
ROM size	2-4KB	0.5-6MB	16MB
RAM size	128B	128KB	72KB
Color depth	8 bit	16 bit	16 bit
Screen Size	160x210	256x224 or 512x448	320x224
Possible buttons combinations	18	over 720	over 100

2.4 ENVIRONMENT CHALLENGES

Integrating SNES with RLE presents new challenges to the field of RL where visual information in the form of an image is the only state available to the agent. Obviously, SNES games are significantly more complex and unpredictable than Atari games. For example in sports games, such as NBA,

[2] http://stella.sourceforge.net

[3] http://www.snes9x.com

[4] https://github.com/ekeeke/Genesis-Plus-GX

[5] https://github.com/nadavbh12/Retro-Learning-Environment

in which the player (agent) controls a single player, while all the other nine players' behavior is determined by pre-programmed agents, each exhibiting random behavior. Moreover, unlike Atari that consists of a maximum of 18 actions, SNES actions space may be larger than 700. Furthermore, the background in SNES is very rich, filled with details which move locally or across the screen. Finally, we note that SNES utilized the first 3D games which still provide significant challenges.

3 EXPERIMENTS

We performed several experiments in which we examined several SNES games by running the Deep Q-Learning (DQN) (Mnih et al., 2015) algorithm as well as two of its extensions: Double DQN (Wang et al., 2015) and Dueling Double DQN (Van Hasselt et al., 2015). The evaluation methodology that we used for benchmarking is the popular method proposed by (Mnih et al., 2015). While the screen dimensions in SNES are larger than those of Atari, in our experiments we maintained the same pre-processing as suggested by Mnih et al. (2015) (i.e., downscaling the image to 84x84 pixels and converting to gray-scale). To handle the large action space, we limited the algorithm's actions to the minimal button combinations which provide unique behavior. Additional experiments where we utilized both reward shaping and multi-agent training (letting two agents compete one against the other) are available in the full version of our paper.

3.1 RESULTS

A thorough comparison of the four different agents' performances on SNES games can be seen in Figure (1) Only in the game *Mortal Kombat* a trained agent was able to surpass a expert human player performance as opposed to Atari games where the same algorithms have surpassed a human player on the vast majority of the games. A video demonstrating the agent's performance on the game Gradius III can be found at youtu.be/nUl9XLMveEU .

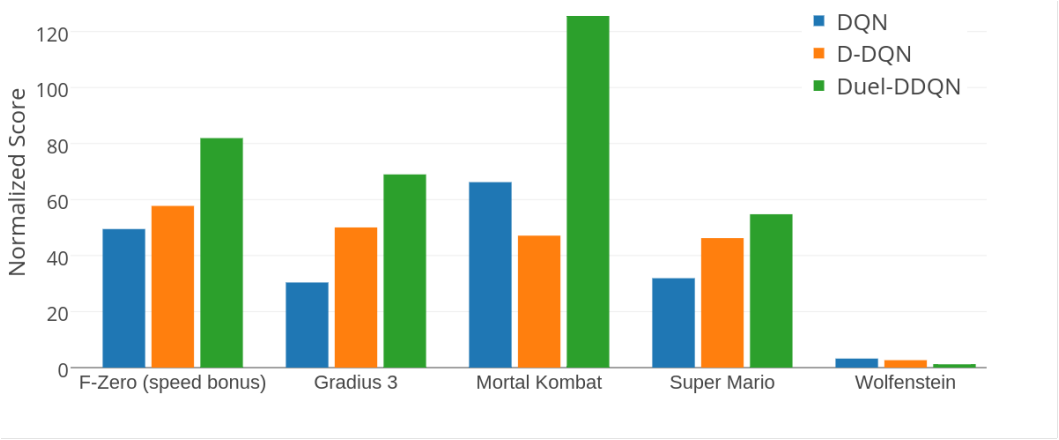

Figure 1: DQN, DDQN and Duel-DDQN performance. 100 represents a human player and zero a random agent.

4 CONCLUSION

We introduced a rich environment for evaluating and developing reinforcement learning algorithms which presents significant challenges to current state-of-the-art algorithms. In comparison to other environments RLE provides a large amount of games with access to both the screen and the in-game state. The modular implementation we chose allows extensions of the environment with new consoles and games, thus ensuring the relevance of the environment to RL algorithms for years to come. The challenges presented in the RLE consist of: 3D interpretation, delayed reward, noisy background, stochastic AI behavior and more. Although some algorithms were able to play successfully on part of the games, to fully overcome these challenges, an agent must incorporate both technique and strategy. Therefore, we believe, that the RLE is a great platform for RL research.

5 ACKNOWLEDGMENTS

The authors are grateful to the Signal and Image Processing Lab (SIPL) staff for their support, Alfred Agrell and the LibRetro community for their help and Marc G. Bellemare for his valuable inputs.

REFERENCES

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

B. Bischoff, D. Nguyen-Tuong, I.-H. Lee, F. Streichert, and A. Knoll. Hierarchical reinforcement learning for robot navigation. In *ESANN*, 2013.

libRetro site. Libretro. www.libretro.com. Accessed: 2016-11-03.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua. Long-term planning by short-term prediction. *arXiv preprint arXiv:1602.01580*, 2016.

H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*, 2015.

Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.