Politecnico di Milano • Honours Programme • November 2018 • CSE Track

State of the Art on: Deep Learning for Video Games AI Development

Emilio Capo, emilio.capo@mail.polimi.it

1. Introduction to the research topic

With the recent advancements in machine learning and, specifically, deep learning techniques, video games applications for AI research are becoming more and more popular, as they prove to be very useful testbeds for general AI algorithms evaluation [1]. At the same time, the need for a step forward in AI development, considering that the videogame industry has now reached an audience comparable to that of music and movies, is strongly perceived by both game developers and players. The former aim at building more complex and entertaining experiences, which in terms brings the latter to push for a higher believability and coherency of the game worlds they explore. Thanks to the amount of data that is being gathered on many video games, experiments showed how promising deep learning proves to be in this direction [2].

Some of the most important venues of the field are the IEEE Conference on Computational Intelligence and Games (CIG), the AAAI Artificial Intelligence and Interactive Digital Entertainment (AIIDE) and the IEEE Transactions on Games (ToG) (previously IEEE Transactions on Computational Intelligence and AI in Games). These are cited in [3] by Georgios N. Yannakakis and Julian Togelius, two prominent games AI researchers.

Most other influential venues are cited in [4].

1.1. Preliminaries

The field is strongly based on statistics and on the mathematical model known as Neural Network (NN), a graph model that mimics the structure of the human brain, using a series of nodes (neurons), organized in layers that exchange information with each other. The number of layers represents the abstraction power of the neural network. Deep learning is based on the idea of using a high number of layers to improve the network abstraction capabilities. After its first applications in Convolutional NN (CNN), largely used in image analysis, a temporal component was added in Long Short-Term Memory NN (LSTM) [5], where some memory of previously elaborated data is stored. The resulting network is then trained according to four main paradigms:

- *Supervised Learning* (SL), which requires a large and representative set of examples (input-output pairs) to learn from;
- *Adversarial Learning* (AL), which exploits the aid of a second neural network that is trained to generate more and more diverse examples for the first one;
- *Evolutionary Learning* (EV), which lets the most efficient network arise by sequentially evaluating the fitness and combining the members of a population;

- *Reinforcement Learning* (RL), which stimulates learning through a reward system, where effective actions are rewarded, while ineffective actions are punished;

The latter is mostly used in games AI in the form of Q-Learning. Networks that are trained this way are referred to as Deep Q Networks (DQN) [6].

Implementation tools include, on one hand, general libraries and toolkits *(TensorFlow, PyTorch, OpenAI Gym*) for artificial intelligence and machine learning applications. Integrated with many game engines (*Unity*, *Unreal Engine*), libraries allow researchers to rapidly build controllable environments for their experiments.

On the other hand, there are frameworks that offer interesting learning environments, together with the necessary APIs to interact with them. Their extensive use consolidated them as benchmarks for games AI applications. Some of the most famous examples are:

- *Arcade Learning Environment* (ALE) [7], an object-oriented environment that offers more than 50 different Atari video games to develop AI agents on; mostly used for General Video Game Artificial Intelligence (GVGAI) applications;
- *VizDoom* [8], a reinforcement learning environment based on the video game "Doom"; focusing the learning process on raw visual data, it is thus suited for deep reinforcement learning applications;
- *TORCS* [9], an AI research platform for car racing agents in a 3D environment, primarily focused on visual reinforcement learning; it offers built-in data structures for neural networks applications;

Other examples are *Ms. Pac-Man* [10], *Project Malmo* [11] and *Brood War API* (BWAPI) [12].

1.2. Research Topic

Artificial intelligence applications in games branch into ten major areas [2]. For deep learning, we mainly identify two of them.

On one hand, the limits arising from traditional artificial intelligence techniques (such as search and planning) in commercial games are becoming more and more evident. Deep learning applications in *Non-Player Character* (NPC) *behaviour learning*, whose focus is to use RL techniques to learn policies/behaviors to efficiently play games, hold the promise of generating more interesting and coherent entities for the players to engage with. This would help create believable agents, that is agents who appear to have human-like characteristics, which would in turn increase the quality of computational narrative.

On the other hand, video games provide enough diversity in simulation environments to constitute an efficient first step towards a human-level domain-general artificial intelligence. The subfield of *General game AI* has thus the objective to create agents that are capable of playing different games. This would allow AI to become detachable from games and, eventually, it would be possible to build AI engines that can be used for different games, similarly to how game engines are used. The focus is currently on Atari games available on the Arcade Learning Environment.

Other areas where deep learning is being used as an aid are player modelling, Procedural Content Generation (PCG) and AI-assisted tools for game design.

2. Main related works

2.1. Classification of the main related works

The current works on deep learning applied to games can be mainly classified along two dimensions [13]. One is that of the learning paradigm used to train the network; the other is the game genre we are considering for our application.

2.1.1 Classification per learning paradigm

There are three main learning paradigms used in this field: supervised learning, reinforcement learning and evolutionary learning.

The *supervised learning* paradigm is mainly used to let agents learn a behaviour from data recorded either from human players or from algorithms that play the game well. The network can also be a support for other techniques by predicting the next most probable game state given the current one. Other algorithms can then determine the best action to take. The main issues related to this paradigm are size and representativeness of the data set at hand. Without a sufficient amount of data coming from a sufficiently wide portion of the state space, supervised learning cannot be efficiently applied.

The *reinforcement learning* paradigm stimulates learning through environment exploration and interaction. This paradigm can be easily applied to games by modeling them as environments in a RL setting, where players have a certain number of actions at their disposal, whose combination determines their success. The reward function can often be identified by the game score itself. The issue with its application to games is that it must rely on model-free techniques, such as Q-learning and SARSA, because a full exploration of such complex environments in reasonable time is computationally unfeasible.

The *evolutionary learning* paradigm has been used for different tasks in games. It was used as a support for search strategies to evaluate hypothetical future game states, applied to games such as Checkers and Go. Differently, it was used to directly select actions for car driving agents and shooter games. Its limitations mainly concern learning from high-dimensional data, whom evolution can't seem to be able to deal with, and its application to general video games AI, which has been seriously understudied, in spite of evolution generalization capabilities.

2.1.2 Classification per game genre

Performing an analysis by game genre reveals how some genres are preferred to others in some specific applications.

For instance, *arcade games* have been primarily used for General Video Game AI (GVGAI) applications, thanks to their relatively simple structure and the variety they offer. These games, available on the ALE framework, all share some characteristics, such as a continuous-time progression and either continuous or discrete space movement, but also differ in the skills required to play them, sometimes significantly. Some games require fast reaction and timing, some others require prediction of the behaviour of some elements in the game, and some even require maze navigation or long-term planning.

In *racing games*, the player has to control a vehicle, usually with the objective to reach a given goal in the shortest time. The main purpose of DL applications to this genre is to create an agent that can efficiently drive the car, providing a continuous output, that is the steering output, given as input either sensor data or raw pixels (or even both). Sometimes, learning is complicated by resource management (fuel, wheels durability) or by adversarial contexts (driving against other agents). The main platform used for this kind of experimentation is TORCS.

First-Person Shooters (FPS) often offer partially-observable 3D environments for developing visual RL agents. Featuring an over-the-shoulders viewpoint, their primary challenge is fast reaction to enemy sighting, but they also require efficient tridimensional space exploration, as well as prediction of enemy behaviour and resource location. The main platform for such experiments is VizDoom.

Offering an even harder challenge than their board counterparts, *Real-Time Strategy* (RTS) *games* require careful planning and managements of different kinds of units at the same time, in order to obtain diversified resources and, eventually, defeat the opponent. These games feature an enormous branching factor, as well as long planning to produce effective results. Most efforts in this direction are applied to the BWAPI framework.

Other game genres are *open-world games*, which require management of large freedom of action and selecting meaningful goals, and *sports games*, whose focus is largely on cooperation and coordination.

2.2. Brief description of the main related works

We will now proceed to a quick analysis of the main related works [13], specifying how they fit in the above-defined classification.

The most used topology across all genres is CNN, meaning that most works are based on a visual input of the game state. This topology is usually combined with a *RL paradigm* (Q-learning, specifically), resulting in a DQN architecture. Its main feature is experience replay, where some experiences are sampled from a batch during update, allowing to learn from past and uncorrelated experiences. From its first basic application [6], DQNs have been extended thanks to its widespread use in *arcade games*. A recurrent component (DRQN) [16], considering some time information, showed to benefits the network in case of partially-observable games. A distributed implementation outperformed the standard one in most games [17]. Double DQNs [18] solved the issue of action-value function overestimation. A technique called prioritized experience replay allowed to sample more frequently the samples that turned out to be more significant, which improved both standard and double DQNs [19]. Dueling DQNs separately estimate the value function and an action-advantage function [20]. Bootstrapped DQNs improved the exploration policy and, thus, the training time required to train multiple networks [21]. Some multi-threaded asynchronous variances have also been tested (A3C) [22]. Another algorithm, called UNREAL, exploits a replay memory from which it learns auxiliary tasks and pseudo-reward functions [23]. NoisyNets replace the ε-greedy policy with the exploration of a noisy version of the network, improving both DQN and A3 [24]. Finally, the Rainbow technique combines different of the above-mentioned enhancements, achieving a mean score higher than any of the enhancements individually [25].

In *racing games*, we can identify two main paradigms for vision-based autonomous driving: learning image-action mapping directly or mediated learning through sensor data. An in-between approach is direct perception, where images are mapped to data that is then used to make decisions [26]. This method can produce systems that can drive in diverse environments and even generalize to real images. DQNs cannot be applied to continuous domains such as that of racing games, thus other *RL techniques* such as policy gradient method are preferred. These include actor-critic [27] and Deterministic Policy Gradient (DPG) [28]. Deep DPG and A3C have been applied to TORCS showing quite positive results [22] [29].

In *FPS*, human-level capabilities were reached using a CNN with max-pooling and fully connected layer trained with a DQN applied to VizDoom [30]. Another successful method used a CNN trained with A3C [31]. Significant improvements in exploration, an extensively studied important skill for FPS, were reached using a combination of CNN and LSTM trained with A3C [32]. Applications of UNREAL, specifically to gathering tasks in Open Arena, showed even better results than A3C [23].

Applications to *RTS games* mainly focus on subproblems, given the complexity of controlling different agents at the same time in a world without any in-game scoring system. Through the BWAPI framework, the problem of micromanagement was tackled using CNN combined with MCTS, exploiting combat damage as a reward function [15] [33]. Other applications include Independent Q-Learning (IQL), which efficiently managed the multi-agent RL problem [34]; the Multiagent Bidirectionally-Coordinated Network (BiC-Net), based on a recurrent NN [35]; Counterfactual Multi-Agent (COMA) policy gradients, which is an actor-critic method [36]. Some applications in macromanagement, such as prediction of human strategy choices using *SL techniques*, efficiently enhanced in-game bots [37].

2.3. Discussion

Currently, deep learning techniques applied to video games face many open challenges.

From an industry viewpoint, the task of designing a deep learning AI that requires a reasonable amount of computational resources and can be easily integrated is further complicated by business and revenue requirements. A black-box technology that is so unpredictable can easily produce unwanted NPC behaviours that could ruin the game experience, as well as complicate the Q.A. fine-tuning and testing process, hardly allowing for specific skill level tuning. It is thus not surprising that the industry is still exploiting as much as possible more stable and reliable methods, such as search and planning techniques.

The problem of meeting the industry feasibility requirements is mainly related to the complexity of defining an input that is at the same time informative and simple enough to be managed and produce an output in a reasonable amount of time. Similarly, high dimensional output space could be tackled by designing high-level actions.

On the other hand, having a large amount of high-quality data to train these agents can often make the difference in supervised settings. For reinforcement learning, an efficient design of reward function, as well as representative episodes to learn from, constitute the basis for agent training. When dealing with games with very sparse rewards, such as Montezuma's Revenge, RL through a hierarchical implementation of DQL [14] barely reached 10% of the performances of a human-expert player.

Finally, in GVGAI great progress was made by introducing progressive NNs, which allow to learn to play new games without forgetting old ones. However, this method still requires one separate network per task, so further future developments could improve this point.

References

[1] J. Togelius. AI Researchers, Video Games Are Your Friends! Volume 669 2017, Springer.

[2] V. Mnih *et al.* Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

[3] G. N. Yannakakis and J. Togelius. A Panorama of Artificial and Computational Intelligence in Games. 2014, IEEE Transactions on Computational Intelligence and AI in Games.

[4] http://www.kmjn.org/game-rankings/

[5] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Advances in neural information processing systems, pages 3338–3346, 2014.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[7] ALE Github Repository: https://github.com/mgbellemare/Arcade-Learning-Environment

[8] VizDoom Official Website: http://vizdoom.cs.put.edu.pl/

[9] TORCS Official Website: http://torcs.sourceforge.net/

[10] Ms. Pac-Man Competition Website: http://www.pacmanvghosts.co.uk/

[11] Project Malmo page on Microsoft Website: https://www.microsoft.com/en-us/research/project/project-malmo/

[12] BWAPI Github Repository: https://github.com/bwapi/bwapi

[13] N. Justesen, P. Bontrager, J. Togelius and S. Risi. Deep Learning for Video Game Playing.

[14] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In Advances in Neural Information Processing Systems, pages 3675–3683, 2016.

[15] N. A. Barriga, M. Stanescu, and M. Buro. Combining strategic learning and tactical search in real-time strategy games. arXiv preprint arXiv:1709.03480, 2017.

[16] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. arXiv preprint arXiv:1507.06527, 2015.

[17] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al. Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296, 2015.

[18] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In AAAI, pages 2094–2100, 2016.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.

[20] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581, 2015.

[21] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped DQN. In Advances In Neural Information Processing Systems, pages 4026–4034, 2016.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, 2016.

[23] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397, 2016.

[24] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, et al. Noisy networks for exploration. arXiv preprint arXiv:1706.10295, 2017.

[25] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. arXiv preprint arXiv:1710.02298, 2017.

[26] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision, pages 2722– 2730, 2015.

[27] T. Degris, P. M. Pilarski, and R. S. Sutton. Model-free reinforcement learning with continuous action in practice. In American Control Conference (ACC), 2012, pages 2177–2182. IEEE, 2012.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 387–395, 2014. [29]

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[30] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Ja´skowski. Vizdoom: A Doom-based AI research platform for visual reinforcement learning. arXiv preprint arXiv:1605.02097, 2016.

[31] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In Submitted to Intl Conference on Learning Representations, 2017.

[32] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673, 2016.

[33] M. Stanescu, N. A. Barriga, A. Hess, and M. Buro. Evaluating realtime strategy game states using convolutional neural networks. In IEEE Conference on Computational Intelligence and Games (CIG 2016), 2016.

[34] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the tenth international conference on machine learning, pages 330–337, 1993.

[35] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. arXiv preprint arXiv:1703.10069, 2017.

[36] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926, 2017.

[37] N. Justesen and S. Risi. Learning macromanagement in StarCraft from replays using deep learning. In Computational Intelligence and Games, 2017. CIG 2017. IEEE Symposium on. IEEE, 2017.