Training Agents to Play Modern Games: Challenges and Opportunities

Yunqi Zhao, Ahmad Beirami, Mohsen Sardari, Navid Aghdaie, Kazi Zaman
Electronic Arts Digital Platform – Data & AI
209 Redwood Shores Pkwy, Redwood City, CA 94065
{yuzhao, abeirami, msardari, naghdaie, kzaman}@ea.com

Abstract

Games have been instrumental to advancing artificial intelligence (AI). In particular, deep reinforcement learning has shown great success in achieving super-human level performance on Atari games, the game of Go, and Dota 2. Training AI agents in modern complex games brings many challenges, such as massive computational requirements and huge action/state space, rendering many of the techniques that led to recent success stories inapplicable. We consider training AI agents to play a modern mobile game and report preliminary progress toward human-like behavior.

Introduction. The human-game interaction dates hundreds of years back to the emergence of strategic games, such as Shatranj (Chess) and Weiqi (Go), which were developed as a way of bringing strategic thinking to the military, and evolved over multiple centuries to better engage human players.

In addition to challenging humans to strategic thinking, games have been instrumental to advancing artificial intelligence (AI), particularly reinforcement learning (RL). IBM Deep Blue was the first AI agent who beat the chess world champion, Gary Kasparov [1]. Two decades later, DeepMind demonstrated that deep neural networks combined with Monte Carlo tree search (MCTS) [2, 3] could lead to AI agents that play Go at a super-human level [4], and solely via self-play [5, 6]. More recently, OpenAI showed that AI agents could learn to cooperate at a super-human level in Dota 2 [7].

The impressive recent progress on RL for solving games is partly due to the advancements in processing power and AI computing technology.[1] Further, deep Q networks (DQNs) have emerged as a general representation learning framework combined with Q function approximation without need for task-specific feature engineering [9]. The design of a DQN and setting the hyperparameters is still a daunting task. In addition, it takes hundreds of thousands of state-action pairs for the agent to reach human-level performance. Applying the same techniques to modern games would require obtaining and processing even more state-action pairs, which is infeasible in most cases because speeding up the game engine may not be possible and game state may be difficult to infer from the frame buffer.

On modern strategy games, DeepMind and Blizzard showed that existing techniques fall short even on learning the rules of StarCraft II [10]. On the other hand, breaking the sate space and action space into a hierarchy of simpler learning problems has shown to be promising [10, 11] (cf. [12]). Training agents to play modern computer games, particularly in the design stage, poses some novel challenges:

1. the game state space is huge, with continuous attributes, only partially observable to the agent;
2. the set of available actions is huge and unknown to the agent rendering MCTS infeasible;
3. the game itself is dynamic in the design stage and several attributes may change between builds;
4. the games are designed to last millions of ticks leading to potentially long episodes, and the way the player engages with the game environment makes an impact on the gameplay strategy;
5. multiple players may interact in conflict or cooperation leading to an exploding state space; and
6. the goal of the agent could be to "engage" with real human players or play like humans rather than to win the game making it non-trivial to design a proper rewarding mechanism.

[1] The amount of AI compute has been doubling every 3-4 months in the past few years [8].

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Problem setup. We consider a modern mobile game by Electronic Arts (EA) designed to engage many players for months exhibiting all of the above challenges. Our primary goal is to train agents that play the game like human players do. To alleviate the huge state space and the dynamic nature of the game in design space, we move away from processing frame buffers with the hope of training re-usable agents. Instead, we let the game engine pass information that the player can infer in the game, such as the inventory, resources, buildings, and the state of neighboring players ($\sim$50 continuous and $\sim$100 discrete states). The set of available actions consists of $\sim$25 functions taking 5-20 arguments. This set is unknown to the agent and only a handful of the actions are valid (available) at each step. Note that the game server is capable of validating whether a submitted action is available, while it is not straightforward to encode and pass the set of available actions (along with their permissible arguments) to the agent at every time step. While the problem of a huge state space could be dealt with (cf. [13, 14, 15]), these techniques are not directly applicable due to the huge and unorganized action space in this setup. We study game progression while taking only valid actions. Unfortunately, the set of the valid actions (or a distribution on it) is not fully determined by the current observation, and hence, we deal with a partially observable Markov decision process (POMDP).

Method. In this EA game, we show progress toward training an AI agent that takes valid actions, like human players. We create a universal interface between the Gameplay Environment (which encapsulates the game) and the AI Learning Environment (where the agent is trained). The interface extends OpenAI Gym [16] and supports actions that take arguments, which is necessary to encode functions and is consistent with PySC2 [10, 17]. In addition, our training pipeline enables creating new players on the game server, logging in/out an existing player, and gathering data from expert demonstrations. We adapt Dopamine [18] to this pipeline to make DQN [9] and rainbow [19] agents available for training in the game. In particular, we add support for more complex preprocessing other than stacking frame buffers. We use a network with two fully connected layers and ReLU activation.

We create an episode by setting an early goal state in the game that takes a human player $\sim$10 minutes to reach. Note that we are working on extending the setup to learn long-term strategies coupled with engagement with the game. We let the agent submit actions to the game server every second. We reward the agent with '+1' when they reach the goal state, and with '-1' when they take an invalid action, '0' when they take a valid action, and '-0.1' when they choose the "do nothing" action. The game is such that at times the agent has no other valid action to choose, and hence they should choose "do nothing" but such periods do not last more than a few seconds. We intend to modify the rewarding mechanism by inferring traits from the massive player gameplay data at our disposal.

Preliminary results. We consider two different versions of state space. The first is what we call the "complete" state space. The complete state space contains information that is not straightforward to infer from the real observation in the game and is only used as baseline for the agent. The second is what we call the "augmented" observation space where the state space only contains what the agent would be able to learn and retain knowledge about from the actual observations in the game. We train the following agents: **Agent 1** is a DQN agent with complete state space; **Agent 2** is a rainbow agent with complete state space; and **Agent 3** is a rainbow agent with augmented observation space.

The result of the training is shown in Figure 1. As can be seen, the rainbow agent converges to a better performance level while with more training episodes compared to the DQN agent. We also see that the augmented observation space makes the training slower and also results in a worse performance on the final strategy. We are currently (a) shaping the reward for human-like behavior, (b) studying the impact of the neural network structure and hyperparameters on the performance of DQN and rainbow agents, (c) augmenting training with expert demonstrations, and (d) exploring a systematic solution for augmenting the observation space.

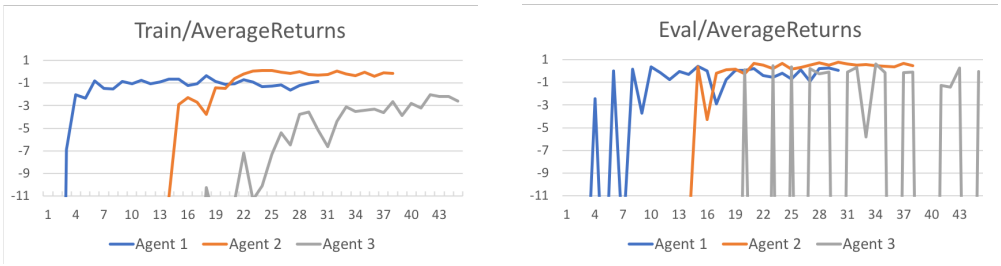

Figure 1: Average cumulative reward (return) in training and evaluation for the agents as a function of the number of iterations. Each iteration is worth $\sim$20 minutes of gameplay.

References

[1] IBM Deep Blue. [Online] http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue.

[2] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[3] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[5] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[7] OpenAI Five. [Online, June 2018] https://openai.com/five.

[8] AI and Compute. [Online, May 2018] https://blog.openai.com/ai-and-compute.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[10] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[11] Zhen-Jia Pang, Ruo-Ze Liu, Zhou-Yu Meng, Yi Zhang, Yang Yu, and Tong Lu. On reinforcement learning for full-length game of starcraft. *arXiv preprint arXiv:1809.09095*, 2018.

[12] Marc Toussaint, Laurent Charlin, and Pascal Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *UAI*, volume 24, pages 562–570, 2008.

[13] Jesse Hoey and Pascal Poupart. Solving POMDPs with continuous or large discrete observation spaces. In *IJCAI*, pages 1332–1338, 2005.

[14] Matthijs TJ Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of artificial intelligence research*, 24:195–220, 2005.

[15] Josep M Porta, Nikos Vlassis, Matthijs TJ Spaan, and Pascal Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7(Nov):2329–2367, 2006.

[16] [Online] https://gym.openai.com.

[17] [Online] https://github.com/deepmind/pysc2.

[18] Marc G. Bellemare, Pablo Samuel Castro, Carles Gelada, and Saurabh Kumar. [Online, 2018] https://github.com/google/dopamine.

[19] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.