

A Deep Reinforcement Learning Agent for General Video Game AI Framework Games

Eddah K. Sure

College of Information Science & Technology
Donghua University
Shanghai, China

Xiaofeng Wang

College of Information Science & Technology
Donghua University
Shanghai, China

Abstract—The General Video Game AI (GVGAI) software framework, which was originally created for General Game Playing (GGP), can be connected via the OpenAI Gym interface for the purpose of testing machine learning agents on a large number of 2-dimensional arcade-type games. In this paper, we present a Proximal Policy Optimization (PPO) based deep reinforcement learning game-playing agent. Using the OpenAI Gym interface, we highlight how this agent performs on a number of randomly selected GVGAI games.

Keywords—reinforcement learning, artificial intelligence, video games, general game playing

I. INTRODUCTION

For decades, games have played a significant role in Artificial Intelligence research as games afford researchers wittults that are easily quantifiable, that is, whether rules were followed, a game level was cleared successfully, points scored or a game was won. Game playing, which requires an agent to make a series of decisions towards winning is seen as a facsimile of the human thought process as they are designed in a way that challenges our human cognitive abilities, therefore, providing good benchmarks for artificial intelligence. So far, research on artificial intelligence for games has been focused more toward the design and building of agents for playing the games.

Board games such as Chess and Go have been the centre of research in the field of artificial intelligence since its inception. However, over the last decade, video games have increasingly become the preferred domain for testing new deep learning algorithms with significant results on a good number of video games. For example, Atari2600 games on the arcade learning environment (ALE) have received overwhelming attention from the research community with researchers modeling outstanding agents and algorithms including the 2012 Deepmind's deep Q-network [1], Never Give-Up algorithm [2], a reinforcement learning agent for hard exploration games that obtained a 1334.0% human normalized score by doubling the performance of the agent in all hard to explore games and while maintaining a high score on the other games on the Atari2600 games and recently Agent57 [3] the first deep reinforcement learning agent that performs beyond the human standard benchmark on all the 57 Atari2600 games e.t.c.

To provide more platforms for research into intelligent game-playing agents, several competitions and challenges have been held and are still running with much attention focused on game-specific challenges. As a result, significant contributions have been made in specific game research and game-specific competitions, however, given the nature of

most competitions, a good number of solutions tend to over-specialize in specific games as researchers and contestants focus more on beating the challenges. Therefore, providing game-engineered and highly tailored heuristics that work effectively for a specific game used in the challenge. GVGAI aims to tackle this problem by providing a framework that allows for the devising of an algorithm that is capable of playing any random game, even if the agent has not seen or has no knowledge of the given game a priori. This might as well be interpreted as an approximation of General Artificial Intelligence as it gives no room for heuristics dependent on a specific game [4].

The need for an agent that can perform great on multiple tasks brought forth the need for a General Game Playing (GGP) challenge where an artificial intelligence agent has to figure out strategies to play and win more than one game. [5] sought to extend this concept of General Game Playing to include video games, therefore, forming the General Video Game Playing (GVGP) and proposed running a competition where agents are tasked to play video games they have not seen before. To facilitate this they investigated the need for the development of a Video Game Descriptive Language (VDGL) to precisely specify video games to accommodate a competition platform for researchers to tackle the idea of general intelligence.

The very first implementation of the VGDL was a python interpreter by [6] that allowed agents to play original VGDL games. This parsed py-vdgl was later moved onto a java interpreter for the very first GVGAI competition [5]. Since its initiation in 2014, the GVGAI framework has been used as a multi-track challenge. The original framework was used for playing single-player games and planning agents and later on incorporated the two-player games [7, 8], generation of rules and creation of levels [9]. The planning permitted the implementation of statistical planning algorithms including the Monte Carlo Tree Search (MCTS). In Monte Carlo Tree Search, the agents allow for search trees and the sequencing of actions without the need for any previous training on the games. The learning track that allowed agents to be written in java and python languages was later added to tackle the learning problems. The GVGAI competition which initially aimed at designing artificial intelligence algorithms for playing the games has had its use diversified to level generation and to avoid game-specific modeled agents, the framework was designed in such a way that for every competition a completely new set of games is designed. [4] gave an overview of the motivation behind the GVGAI and the opportunities that can be exploited from the framework. So far, researchers have used the GVGAI framework with different approaches to general artificial intelligence with [13]

highlighting several agents submitted for the competition up to 2018. [10] outlines different strengths and weaknesses of generating video playing agents as well as matching game features with AI agents based on their performance.

In 2018, [11] first interfaced the GVGAI software framework into OpenAI as a way to connect this framework to machine learning agents and were able to provide the benchmark results for eight GVGAI games with three different deep reinforcement learning algorithms for each, highlighting the learning performance of the three, that is Advantage Actor-critic (A2C), Deep Q-network (DQN) and Prioritized Dueling DQN (PDDQN) agents on the eight different set of games and as a result. By connecting the GVGAI software framework to OpenAI, [11] created a new deep reinforcement learning challenge for researchers.

II. BACKGROUND

A. The GVGAI framework

The GVGAI framework is an open-source framework created mainly for running general AI competitions. The GVGAI framework proposes a reinforcement learning challenge where researchers can develop and test different artificial intelligence methods on a wide range of video games. Just like Arcade learning Environment (ALE) [12], the GVGAI framework, game rules are not given and the agent is supposed to make moves within milliseconds by interacting with the environment. While ALE contains a fixed finite number of games such as the Atari2600, which have been widely used as a research tool for artificial intelligence for instance [1-3], the GVGAI framework contains multiple games created using the Video Game Description Language (VDGL) which enables the creation and addition of new games in to the framework either manually or automatically.

VGDL is a critical part of the GVGAI as it is the core of game design for the GVGAI framework. The games designed from the original VGDL for the GVGAI framework are limited to arcade games. These are games grid structure based therefore, discretizes states and the actions, for instance, UP, DOWN, LEFT, RIGHT and NIL with no detailed precision, for example, the Aliens game where the agent moves either right or left. To build more games into the framework [13] introduced an update in VDGL with the ability to establish a continuous physics setting. This setting can allow for objects to move in a tuned manner, therefore, widening the range of video games that can be build in the GVGAI framework. Initially, VDGL only allowed for the creation of single-player games up until [7, 8] further modified the language by adding extra parameters to specify the number of players and since then games on the GVGAI framework are either single-player or 2-player games. Depending on how the agents within a game interact and the termination sets describing a given game two-player games can be either cooperative games or competitive games. So far more than 200 games with 5 different levels for each game have been created.

B. OpenAI Gym

A good number of benchmarks have been released including ALE, Mujoco, GVGAI e.t.c. OpenAI gym [14] was created with the main objective of combining the best aspects of these benchmarks in a single software package

that is easily accessible and more convenient. OpenAI gym currently contains a good number of different environments but with a common interface. These environments are versioned in a manner that ensures the outcomes remain meaningful and that each environment is independent of the other therefore an addition of a new environment will not affect the results of an already existing environment and vice versa as the software gets updated and as the number of environments in the software grow over time.

Connecting the environments via OpenAI gym interface allows for interfacing of reinforcement learning agent implementation. Reinforcement learning makes assumptions that there exists an agent within a given environment. At each time-step the agent within an environment takes an action from a set of possible actions and in return it receives a new observation and a reward depending on how good the action taken was and in most cases the algorithm seeks to maximize the agents reward.

In OpenAI Gym an agent's experience within the environment is episodic in that the experience can be broken into a series of episodes. For each episode, the initial state of the agent is sampled randomly. The goal of episodic learning is to maximize the total amount of reward on each episode and to achieve a high level performance with the least number of episodes per game. The performance of a reinforcement learning algorithm can be determined by two metrics, one is the final performance which is given by the average reward per episode after learning and the second metric is the time taken by an algorithm to learn.

C. Reinforcement learning

Reinforcement learning, a category of machine learning that tends to focus on sequential decision-making [15] where learning is based on feedbacks. In reinforcement learning, an agent uses trial and error experience, learns positive behavior in that the agent either modifies or acquires new behavior and skills incrementally. Therefore, reinforcement learning agents necessarily need not to have complete knowledge of the environment they are in and neither do they need to have control over the environment but only need to be capable of collecting information as they interact with the environment.

The general reinforcement learning problem can be simply modeled and presented as a discrete-time stochastic process. The reinforcement learning agent interacts with the environment starting with an initial state S_0 within an environment S , given by $S_0 \in S$, an initial observation $\omega_0 \in \Omega$. At each and every time-step t , an agent chooses an action a_t to perform from a given set of actions A , given by $a_t \in A$. The agent gets back a reward $r_t \in R$, the state transitions to S_{t+1} and observation ω_{t+1} . This given discrete time stochastic control process is considered a Markov Decision Processes(MDP), a control setting proposed by [16] and later on extended by [17].

The Markov Decision Process can be represented by five key elements; S which defines the state, A defines the action space, T defines the transition function, R defines the reward function which is a continuous set of possible rewards and γ is the discounting function. As the agent interacts with the environment it can observe how each action it takes at each time step affects the quality of the reward it receives. The reward associated with each interaction measures its success

or failure of its immediate action, however, it is common to consider discounted sum of the total reward by multiplying it with a discounted factor γ to encourage short term learning in the agent.

Reinforcement learning methods can be classified as either value-based or policy-based or as a hybrid of both. Value-based deep reinforcement learning algorithms aim at building a value function. This value function subsequently lets us to define a policy. Some of the popular value-based algorithms include Q-learning with its variants. [1] attained human-level performance on most of the Atari2600 video games with a DQN agent using neural networks as function approximators with game pixels as the input. Policy gradient methods aim at optimizing the performance objective of an agent, that is, to accumulate a high reward throughout the game episodes by finding a good policy. In some cases, the policy gradient requires an estimate of a value function for a given policy. This can be achieved with an actor-critic architecture [18] in which an actor refers to the policy while a critic refers to the estimate of a value function. In deep reinforcement learning, both the critic and the actor can be represented by non-linear function approximators [19]. In the first illustration of deep learning algorithms in GVGAI games [1] compared the performance of both value-based and policy-based algorithms that is DQN, PDDQN, and A2C.

III. METHODS

A. GVGAI-Gym

The General Video Game AI framework was designed and implemented by teams from New York University [USA] and the University of Essex [UK] to provide a competition framework and a research tool for general game playing. The compressed framework code can either be downloaded from GitHub then manually unzipped or directly cloned from GitHub repository. In order to interface it with OpenAI Gym, we cloned the framework used for the GVGAI 2020 competition. Once it is cloned you have access to the setup and build files that allow you to install the GVGAI environment. Each game on the GVGAI-Gym environment contains 5 different levels, lvl0 – lvl4 and during the interface on OpenAI Gym, each game environment is registered uniquely indicating that it's a gvgai game, the name of the and the level of the game

In the GVGAI framework, agents in the environment have access to the observation of the current game state, a StateObservation object given as a Json object or as a screenshot of the current game state in a png format. At each game tick, the agent receives a new game state observation and either returns an action within 40 microseconds or sends a request to terminate the game. Once the game is finished or terminated the agent can select the next level from the existing 5 levels.

B. Proximal Policy Optimization

Proximal Policy Optimization is an algorithm that was first proposed and implemented by OpenAI. It is a family of policy gradient methods that alternate between the sampling of data through interaction with the environment and the optimizing of a surrogate objective function using stochastic gradient descent [20]. In reinforcement learning, a policy defines a set of instructions an agent should take depending

on state in the environment that the agent is in. The agent learns by calculating the policy gradient and when evaluating the agent we focus on the policy function to establish how well our agent is performing following a certain policy. PPO is a policy optimization algorithm in that at each step there is an update to the existing policy to improve certain parameters of the algorithm. The algorithm ensures that the update is not too large by clipping the update region to a very narrow range with the advantage function giving the difference between the total sum of discounted rewards on a state-action pair and the value of that policy. Mathematically, the Proximal Policy Optimization is given as:

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (1)$$

where θ is the policy parameter, \hat{E}_t is the empirical expectation over time-steps, r_t denotes the ratio of probabilities under the new and old policies respectively also known as the sampling ratio and is used for update. \hat{A}_t is the estimated advantage at a time t and ϵ is a hyper-parameter that denotes the limit of the range within which an update is allowed and is usually given as either 0.1 or 0.2.

The implementation used in this experiment is the Actor-Critic implementation which is also the most common implementation of the PPO algorithm. The Actor-Critic model uses two deep neural networks with the actor taking the action and the critic handling the rewards by determining how good or bad an action taken by the actor is.

Algorithm 1 PPO Actor-Critic implementation

```

Input: initial policy parameters
for iteration = 1,2,..., do
    for iteration = 1,2,..., N do
        run policy  $\pi_{\theta_{old}}$  in environment for T timesteps
        compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_t$ 
        end for
        optimize surrogate L with respect to  $\theta$ 
         $\theta_{old} \leftarrow \theta$ 
    end for

```

One major challenge is that the proximal policy optimization method is hypersensitive to hyper parameter tuning such as the choice of learning rate. The value of hyper parameters tend to significantly affect the performance of a model. Furthermore, in reinforcement learning, models have many parameters. To get an optimal set of these parameters using a grid search is time consuming. Since a metric of accuracy that indicates the performance of a reinforcement learning model does not exist, therefore, we assume that the parameters are independent of one another and can be trained one after another.

TABLE I. TUNED PARAMETER VALUES FOR EACH GAME USED IN THE EXPERIMENT.

Hyper parameter	Game		
	Aliens	DeceptiZelda	Flower
Learning rate	0.0000644	0.00001	0.0000137
Gamma	0.837	0.656	0.666
Gae_Lambda	0.617	0.843	0.562
n_steps	3781	500	7488

Like other reinforcement learning algorithms, PPO contains many parameters including optimization parameters such as the learning rate which controls the speed at which a model learns and ranges between 0 and 1. If the value is too large then it may result in divergence, on the other hand if the value is very small, the training time may be longer. By determining which action lead to a higher or lower reward the discounting factor, Gamma determines how far-sighted an agent can be. If the value of gamma is too small, the agent will focus more on the immediate reward and if the value is too big, the agent will focus on future or delayed reward.

The tuned hyper parameters for the PPO model were trained for 50000 time-steps due to time limitations using PPO algorithm with 4 evaluation episodes and focused on tuning four main parameters with results shown Table I above.

IV. RESULTS AND DISCUSSIONS

In this section, we highlight the performance of the PPO agent on three different games that is Aliens, DeceptiZelda and flower. Our agent was trained separately for each of the three games. Each embedded with a network created from separately tuned parameters for each of the games. Aliens and Flower were trained for 1 million timesteps while DeceptiZelda was trained for up to 300,000 timesteps as we discovered in the course of the training process that our agent was trapped within a path with no room for further exploration just after less than 10,000 timesteps.

A. Aliens

Aliens is similar to the arcade game SpaceInvaders on Atari2600 library. It is a stochastic game designed such that the enemies move in one direction above the agent that is situated at the bottom of the screen. For the agent to win the game, it has to destroy all the aliens above by shooting missiles at them while avoiding the bombs dropped by moving either left, right or staying in the same position. Our agent learns to play and successfully clears all the five levels of the game while accumulating rewards throughout the training process attaining an average score of 72.

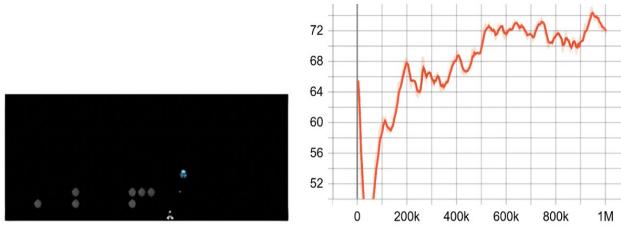


Fig. 1. From left to right a screenshot of the game Aliens at a given game state and the training graph of mean rewards against the time-steps.

B. DeceptiZelda

DeceptiZelda is a deceptive type of game. It is an adventure game where the agent is challenged by presenting it with two paths. One path leading to a quick easy victory, while the other path leading to a much larger reward only if the agent can successfully overcome the moving enemies along the path. The first level of the game gives the agent an option of a quick victory which involves the agent moving down then right and collecting the door key, and therefore, exiting almost immediately without any encounter with the moving enemies. Another choice is for the agent to follow a

path that leads to a region with moving enemies that the agent must fight against to reach an alternative high reward. The agent clears the level by taking the easy quick victory. The first level of DeceptiZelda is small and the agent quickly learns an optimal strategy and once the agent selects the path it becomes trapped in that chosen path such as in our case. The agent figures out the quickest way to earn 2 points along the path of least resistance and therefore does not further explore the rest of the environment.

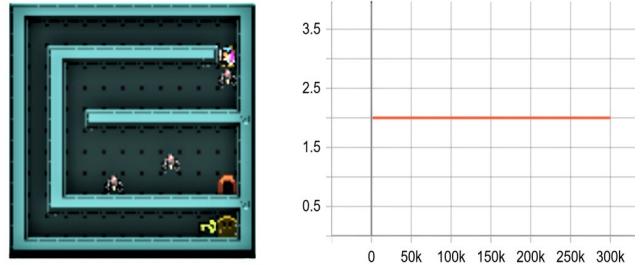


Fig. 2. From left to right a screenshot of the game DeceptiZelda at a given game state and the training graph of mean rewards against the time-steps.

C. Flower

Flower is a game designed in such a way as to reward an agent by offering the agent an opportunity to collect a different amount of rewards ranging from 0 to 10 depending on the size of the flower which is determined by the waiting duration. Seeds are available for the agent to collect but eventually, these seeds grow into flowers with their point values growing along with them. Once a flower is collected another begins to grow as soon as the agent leaves the space from which the flower was collected. The optimal strategy in this game is to wait and allow the flower to grow to maturity by choosing a picking cycle among the spaces that is, the agent learns that it is better to collect a mature flower, 10 points than collecting seeds for 0 points.

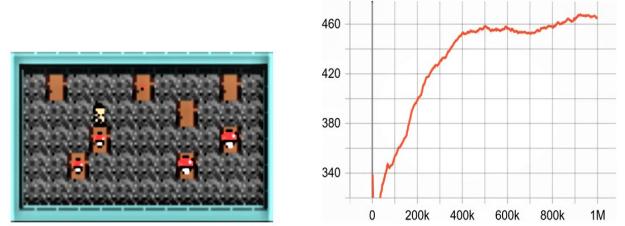


Fig. 3. From left to right a screenshot of the game Flower at a given game state and the training graph of mean rewards against the time-steps.

V. CONLUSION

For development of real-time general video game playing agents, just like Arcade Learning Environment, GVGAI provides suitable testbed. Over time Atari games have become one of the most commonly used benchmarks in reinforcement learning. GVGAI games offer the same benchmarks and from previous and ongoing research on artificial intelligence, deep learning agents have proven to perform exceptionally well in video games. In 2018, [11] yet another deep reinforcement learning challenge was initiated from the GVGAI framework. They provided the very first of benchmark results on eight of the GVGAI game including Aliens using three different algorithms. Our Proximal Policy Optimization based agent performed relatively well attaining a score of 72 in comparison with their three agents, with their

DQN agent achieving a score of 75, PDDQN attained a score of 72.6 and the A2C agent achieving a score of 77 all trained up to 1 million timesteps.

REFERENCES

- [1] V. Mnih et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529-533, 2015.
- [2] A. P. Badia et al., "Never give up: Learning directed exploration strategies," arXiv preprint arXiv:2002.06038, 2020.
- [3] A. P. Badia et al., "Agent57: Outperforming the atari human benchmark," in International Conference on Machine Learning, 2020: PMLR, pp. 507-517.
- [4] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, and S. M. Lucas, "General video game ai: Competition, challenges and opportunities," in Thirtieth AAAI conference on artificial intelligence, 2016.
- [5] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 195-214, 2019.
- [6] T. Schaul, "A video game description language for model-based or interactive learning," in 2013 IEEE Conference on Computational Intelligence in Games (CIG), 2013: IEEE, pp. 1-8.
- [7] R. D. Gaina et al., "The 2016 two-player gvgai competition," *IEEE Transactions on Games*, vol. 10, no. 2, pp. 209-220, 2017.
- [8] R. D. Gaina, D. Pérez-Liébana, and S. M. Lucas, "General video game for 2 players: Framework and competition," in 2016 8th Computer Science and Electronic Engineering (CEEC), 2016: IEEE, pp. 186-191.
- [9] A. Khalifa, D. Perez-Liebana, S. M. Lucas, and J. Togelius, "General video game level generation," in Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, pp. 253-259.
- [10] P. Bontrager, A. Khalifa, A. Mendes, and J. Togelius, "Matching games and algorithms for general video game playing," in Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference, 2016.
- [11] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game ai," in 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018: IEEE, pp. 1-8.
- [12] 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*, vol. 47, pp. 253-279, 2013.
- [13] D. Perez-Liebana, M. Stephenson, R. D. Gaina, J. Renz, and S. M. Lucas, "Introducing real world physics and macro-actions to general video game AI," in 2017 IEEE Conference on Computational Intelligence and Games (CIG), 2017: IEEE, pp. 248-255.
- [14] G. Brockman et al., "Openai gym," arXiv preprint arXiv:1606.01540, 2016.
- [15] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," arXiv preprint arXiv:1811.12560, 2018.
- [16] R. Bellman, "Dynamic programming and stochastic control processes," *Information and control*, vol. 1, no. 3, pp. 228-239, 1958.
- [17] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834-846, 1983.
- [18] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.
- [19] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in International conference on machine learning, 2016: PMLR, pp. 1928-1937.
- [20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.