Deep Reinforcement Learning in Match-3 Game

Ildar Kamaldinov
National Research University Higher School of Economics,
Moscow, Russia
irkamaldinov@edu.hse.ru

Ilya Makarov
National Research University Higher School of Economics,
Moscow, Russia
iamakarov@hse.ru

I. Abstract

***Abstract*—An increasing number of algorithms in deep reinforcement learning area creates new challenges for environments, particularly, for their comprehensive analysis and searching application areas. The key purpose of this article is to provide an extensible environment for researches. We consider a Match-3 game, which has simple gameplay, but challenging game design for engaging players. The article provides metrics for evaluation of agents and corresponding baselines in different scenarios.**

***Keywords*—Deep Reinforcement Learning; Match-3 Video Game; Game Environment; Deep Learning**

II. Introduction

Releases of new environments that would be suitable for reinforcement learning tasks are much rarer than releases of new algorithms. In addition to limiting the possibilities for experimentation and testing of algorithms, this may limit researchers in their search for new algorithms, because in developing them, researchers keep in mind the existing set of environments or tasks from the real world. Because of this, developing new environments is an important task, especially if these environments are connected to the real world. Environments in which it is possible to conduct experiments with a high level of control are in high demand to put forward hypotheses about the algorithm and then test them.

Match-3 is a tile-matching game where a player manipulates tiles on a board in order to make them disappear according to a matching criterion. In many tile-matching games, that criterion is to place a given number of tiles of the same type so that they adjoin each other. Match-3 games become very popular last years, they are top grossing in AppStore and Google Play, their success is due to the simple rules and a short session. Such games are not demanding on player's skills, but level-design is a big challenge for developers because they should implement new levels as soon as possible. Besides "beauty" and "novelty" of new levels, the difficulty is the key variable in players retention and conversion to payers.

As the complexity of the level is necessary to know before its official launch there are so-called playing departments that test the complexity of the level. The duration of the test is a few days. The main alternative to this is a simulation [1], which takes hours, and the game does not necessarily look like a game of real people. Taking into account the described disadvantages of special departments and simulations, the solution may be reinforcement learning.

978-1-7281-1884-0/19/$31.00

The first contribution is an open-source environment with *gym* [2] interface which is easy to use and default 30 levels for experiments. The second contribution is providing baselines of state of the art algorithms performance in model-free setting on the default levels and their augmented versions.

III. Reinforcement Learning Overview

A. General Overview

Two key characters of RL are an agent and an environment. The agent can interact with the environment, which can change due to the agent's actions or on its own.

Every step the agent observes some state (s_t) of the environment (not necessary entire state is accessible). For the given state agent can choose any action (a_t) from action space leading to a new state of the environment (s_{t+1}) and giving some reward (r_t). These four objects compose transition (s_t, a_t, s_{t+1}, r_t).

The sequence of states and actions is a trajectory:

$$\tau = (s_0, a_0, s_1, a_1, ...), \tag{1}$$

where s_0 is sampled from a state-start distribution. To choose the action for a given state, the agent uses a policy, which can be deterministic or stochastic. In deep reinforcement learning we deal with parameterized policies:

$$a_t = \mu_\theta(s_t) \tag{2}$$

$$a_t = \pi_\theta(\cdot|s_t) \tag{3}$$

The goal of the agent is to maximize cumulative reward over a trajectory. Finite-horizon undiscounted return is the sum of rewards over a fixed number of steps, on the contrary infinite-horizon discounted return is the sum over an infinite number of steps, but each reward is discounted by factor γ:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{4}$$

The discount rate determines the present value of future rewards: a reward received t time steps in the future is worth only γ^t times what it would be worth if it were received immediately. Mathematically, if $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence is bounded [3].

B. Application to Match-3

For setting reinforcement task for Match-3 game we should describe key concepts: state, action space, trajectory, reward, and return. But we will start from the description of a level, which is specific to the environment.

a) Level: The main element of the game is the level, which is a board template and the number of shapes on the board. The template is a map which allows determining height and width of the board and coordinates of the fixed shapes. After filling the template with different shapes (number of shapes is an important part of the level) we get a final board.

The dependence of the level's complexity on these parameters is not always obvious: the bigger the size of the table implies the greater the choice for action, the more fixed figures can both simplify the level and complicate it, the greater the number of different figures will always complicate the level.

b) State: The current state of the game in Match-3 is a game board, which consists of movable and fixed shapes. Size of the board is adjustable, the number of shapes on the board is controllable too. That allows us to manage the difficulty of the game for our agent.

c) Action space: The core mechanic of the game allows to swap two neighbors shapes on the board, diagonal moves are prohibited. Match-3 environment can be classified as model-based if the agent knows possible moves which lead to deleting shapes and as model-free if not, that difference appears as two possible action spaces.

Model-based action space Since the environment is simple it is possible to find successful moves by brute force and delegate choice one of them to an agent.

Model-free action space Most of the actions won't change the state because only successful moves are allowed, therefore in model-free setting agent should find actions. For research purpose agent can try to move fixed figures too, it does not make any effect on the board, but step will be taken into account.

d) Trajectory: A trajectory is a sequence of the game board states and moves of shapes on the board. There is important to emphasize, that the environment in Match-3 game is stochastic because each next state generated randomly from previous state and deleted shapes. A special case is a state when there are no possible moves which will cause deleting shapes, in this case, all the shapes on the board will be shuffled.

e) Reward: For the current state, the action just is taken and the next state of the board reward is the number of deleted shapes. Each action of the agent should lead to reward, because the game mechanic does not allow other moves, therefore key role plays planning. For a model-free setting, it is more reasonable to allow the agent to try to make a wrong move and spend its attempt because otherwise there will be no need to search for a move. An important feature is extra-reward from the environment (cascade matching): after deleting shapes on the board new ones are created and they can form a 3-shaped sequence which leads to deleting, that mechanic allows to agent exploit stochastic nature of the environment.

f) Return: Real mobile or web applications consist of finite-horizon games with different additional goals, for e.g., collect N elements of particular shapes or collect Z shapes in 30 seconds (N steps), but core mechanics remain the same: move shapes in sequences for deleting. A goal of the current implementation of the environment is the sum of reward collected after 100 steps.

IV. Related work

A lot of environments are available in python library `gym` [2], from Atari 2600 to robotics and . The interface of this module is easy to use, it is widely-used software among researches nowadays.

DeepMind Control Suite is a set of continuous control tasks with a standardized structure and interpretable rewards [4]. AI safety gridworlds is a suite of reinforcement learning environments illustrating various safety properties of intelligent agents [5]. [6] is an environment for studying agents in competitive multi-agent game, [7] is another one for multi-agent cooperative game.

[8] is an environment inspired by the human game genre of MMORPGs (Massively Multiplayer Online Role-Playing Games, a.k.a. MMOs), this environment is persistent and supports a large and variable number of agents.

CoinRun environment [9] is another environment which provides a metric for an agent's ability to transfer its experience to novel situations.

Key research in Match-3 agents are King's projects [1], [10], [11], [12]. A big difference in current research and King's is that we are focused on the general approach of playing Match-3 game when King tries to predict average human success rate (AHSR) from metrics of an agent.

The first approach in [1] was based on Monte-Carlo Tree Search with Upper Confidence Bound selection strategy [13], which outperforms heuristics and game testers. The biggest drawback of MCTS is the duration of prediction since simulating of game moves is very time-consuming.

In [10] authors used deep neural network (DNN) to predict next move from gameplay data of MCTS-bot. DNN allows to compute AHSR very fast and test level difficulty real-time by game designers. In [11] instead of MCTS-bot game data authors used actions of real users with convolution neural network, which gives the best performance in AHSR prediction in accuracy and prediction time. Key disadvantages of CNN based method is collecting data, because if a new level is very different from others or contains a new feature neural net will not have good performance on states from that levels, then the only thing that is required for MCTS-based approach is the set of possible moves, moreover for the most simple strategy with random actions only actions space is required. But as soon as state-actions data of real players is collected CNN-based approach is possible.

V. Environment and scenario

For our purposes, we developed a new Match-3 environment with commonly used gym interface [2] which is well known among RL researchers.

The current implementation contains 30 levels of different complexity. Users can add new levels by themselves by creating a template, which will show where to put the fixed figures and the number of different figures on the table.

A. Scenarios

a) Scenario 1. Default levels: In the first scenario, we will test the obtained agents on the whole set of default levels. The table size, in this case, is 9x9, the minimum number of figures on the table is 4, the maximum is 7.

b) Scenario 2. Rotated levels: In the last scenario, in order to evaluate the generalization ability of the trained agents, we will run the same tests on the rotations of default levels. Each level was rotated three times on 90 degrees and then we collected unique templates.

B. Policy tests

Below, we enumerate possible tests of an agent, which acts in the described environment.

a) Cumulative reward: The classic method is calculation mean and median cumulative reward on episodes.

b) First correct move: By sorting the algorithm's outputs (q-values or probabilities of actions) for one state we can find the first successful move to evaluate the ability to search for moves.

c) Reward of first correct move: After finding the first correct move we can accumulate rewards from these actions for additional study. The mean and median reward of successful moves can differentiate actions by worthiness.

Length of the test episode is equal to 100. The number of iterations for the tests is equal to 1000.

VI. Proposed Approach

For describing our approach we start with a representation of input state and then give an architecture of the encoder network.

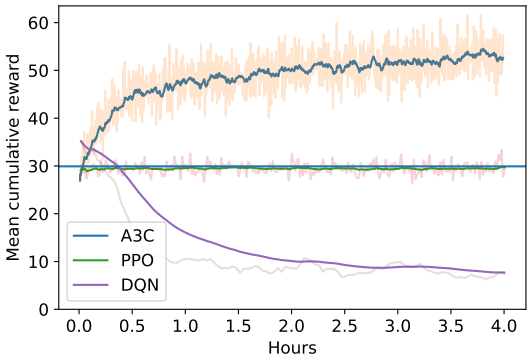
A. Game state

Since Match-3 game board is similar to other board games like chess, shogi and go we adapted input for the network from AlphaZero [14]. Each game has N shapes on the board, therefore our two-dimensional board transformed into to $W \times H \times N$ tensor with own dimension for each type of shape on the board. Due to the fact that each level has its own size, all levels are padded to the maximum board size with fixed figures. For the second scenario with immovable shapes the first dimension of the state-tensor reserved for immovable shapes mask. No other information is provided.

B. Architecture

Deep neural network architecture consists of two main parts: an encoder network and dense layers specified for algorithms (actor and critic parts).

Figure 1. Dynamics of training



a) Encoder: An encoder architecture includes only one depthwise separable convolution [15] with 3x3 filter and ReLU activation. This type of convolution exploits the nature of Match-3 games. For all algorithms feature extractor was trained from scratch.

We trained three reinforcement learning algorithms: DQN [16], PPO [17] and A3C [18]. Intermediate layers specific to each algorithm have ReLU as an activation, size of hidden layers is equal to 128. For detailed implementation see RLlib [19].

VII. Experiments

Only Asynchronous Advantage Actor-Critic showed notable performance, other algorithms work worse than random policy.

The algorithms were trained for 4 hours on 16 CPUs (76437 episodes for A3C, 66840 for PPO and 4560 for DQN). The plot of the dynamics of the average cumulative reward per episode for the second scenario is on the figure 1.

a) First Scenario: An agent trained by A3C algorithm outperforms random policy twice in searching moves, but reward per successful move is almost the same. Asynchronous Actor-Critic Agents was additionally trained for 14 hours, but the mean rank metric was still decreasing. The maximum return remained virtually unchanged during the training for both A3C and PPO and remained at 80, while only an A3C agent made at least one move in the worst case, the others did not make successful moves at all.

Table I. Algorithms performance for Match-3 game in the 1st scenario

	Rank			Reward		
	Mean	Median	Std	Mean	Median	Std
Random	14.92	11	15.41	5.33	3	4.24
DQN	14.05	9	15.44	5.08	3	3.78
PPO	16.93	13	15.58	5.2	3	3.75
A3C	7.5	4	9.61	5.28	3	3.66

b) Second scenario: For the second scenario, we tested policies on rotated board as an examination of generalization. Metrics are on table II. The metrics of the A3C agent are almost the same, that means the agent learned how to play the game instead of learning how to play a limited number of

Table II. ALGORITHMS PERFORMANCE FOR MATCH-3 GAME IN THE 2ND SCENARIO

	Rank			Reward		
	Mean	Median	Std	Mean	Median	Std
Random	15.55	10	16.37	3.88	3	5.23
DQN	14.68	9	16.00	3.84	3	5.14
PPO	16.42	12	15.16	4.15	3	5.29
A3C	7.76	4	10.63	3.77	3	5.38

levels in the game. The maximum cumulative return of agents differs from the first scenario, this time A3C surpassed PPO by 32.5% (155 vs. 117). The minimum cumulative return is the same as in the first scenario.

The most important part of the algorithm is an encoder because it can be used in other algorithms. Most importantly, the feature maps it produces better reflect the behavior of the real human who needs to visually assess the board and then choose the right move. When approaching the task as a model-based environment, we would either not have an encoder (MCTS with UCB1 [20]) or we would have one that evaluates already known moves as it happened when using DQN for Go [21].

Despite the fact that the model-free setting is closer to the human condition, a trained agent does not necessarily have a success rate as the human, so it is necessary to additionally process such indicators. Also, in order to launch new levels, the behavior of certain players clusters may be more important: beginners, experienced, paying, very paying.

VIII. CONCLUSION AND FUTURE WORK

This article presented the open source environment to study deep reinforcement learning similar to the one played by millions of people every day. The article described the Match-3 game in terms of reinforcement learning. Experiments have been carried out in the model-free setting for 30 default levels and their rotations. Future improvements of the Match-3 environment are goal-oriented levels, e.q. achieving some score in 100 moves and deleting a fixed number of specific figures in 100 moves, new figures with a more complex mechanism.

ACKNOWLEDGMENT

The work was supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia. The research continues experiment studies on knowledge representation in Deep RL environments Pong [22] and VizDoom [23].

REFERENCES

[1] E. R. Poromaa, "Crushing Candy Crush," Tech. Rep. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:1093469/FULLTEXT01.pdf

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[3] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[4] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. De, L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller, "DeepMind Control Suite," Tech. Rep., 2018. [Online]. Available: https://arxiv.org/pdf/1801.00690.pdf

[5] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "AI Safety Gridworlds," nov 2017. [Online]. Available: http://arxiv.org/abs/1711.09883

[6] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. Bowling, "The Hanabi Challenge: A New Frontier for AI Research," feb 2019. [Online]. Available: http://arxiv.org/abs/1902.00506

[7] S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, and T. Graepel, "Emergent Coordination Through Competition," feb 2019. [Online]. Available: http://arxiv.org/abs/1902.07151

[8] J. Suarez, Y. Du, P. Isola, and I. Mordatch, "Neural MMO: A Massively Multiagent Game Environment for Training and Evaluating Intelligent Agents," mar 2019. [Online]. Available: http://arxiv.org/abs/1903.00784

[9] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying Generalization in Reinforcement Learning," dec 2018. [Online]. Available: http://arxiv.org/abs/1812.02341

[10] S. Purmonen, "Predicting Game Level Difficulty Us-ing Deep Neural Networks," Tech. Rep., 2017. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:1154062/FULLTEXT01.pdf

[11] P. Eisen, S. F. Gudmundsson, and J. Dowling, "Simulating Human Game Play for Level Difficulty Estimation with Convolutional Neural Networks," Tech. Rep. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:1149021/FULLTEXT01.pdf

[12] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-Like Playtesting with Deep Learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, aug 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8490442/

[13] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *In: ECML-06. Number 4212 in LNCS*. Springer, 2006, pp. 282–293.

[14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," dec 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[15] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," oct 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," jul 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[18] 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," feb 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[19] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," dec 2017. [Online]. Available: http://arxiv.org/abs/1712.09381

[20] C. Browne, E. Powley, D. Whitehouse, S. Lucas, S. Member, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, 2012. [Online]. Available: http://www.incompleteideas.net/609dropbox/otherreadingsandresources/MCTS-survey.pdf

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015. [Online]. Available: http://www.nature.com/articles/nature14236

[22] I. Makarov, A. Kashin, and A. Korinevskaya, "Learning to play pong video game via deep reinforcement learning," *CEUR WP*, pp. 1–6, 2017.

[23] D. Akimov and I. Makarov, "Deep reinforcement learning in vizdoom first-person shooter for health gathering scenario," in *MMEDIA*, 2019, pp. 1–6.