Reward Shaping for Deep Reinforcement Learning in VizDoom

Vitalii Sopov[1], Ilya Makarov[1,2]

[1]HSE University, Moscow, Russia, Pokrovsky boulevard 11, 109028 Moscow, Russian Federation

[2]Artificial Intelligence Research Institute (AIRI), Moscow, Russia, Nizhny Susalny lane 5 p. 19, 105064 Moscow, Russian Federation

Abstract

Reward shaping helps reinforcement learning agents to succeed in challenging tasks when environmental rewards are either sparse or delayed. In this work we propose an approach which combines both information from the game screen and additional information about in-game events to produce an estimation of novelty of the visited states and used behaviors. We use this estimation to motivate the agent to seeking novel experiences and show that our method helps in accelerating learning and reaching better and secures more robust strategies in complex VizDoom scenarios.

Keywords

Deep reinforcement learning, VizDoom, Reward shaping

1. Introduction

Reinforcement learning (RL) in environments with sparse or delayed rewards have traditionally been considered a difficult task to solve. Traditional algorithms such as DQN, DDQN, A2C and others can rarely learn fast enough or converge to satisfying strategies and results without any reward-shaping, which is modification of the reward function to make it less sparse. Such modifications are usually manually made by the creators of algorithms and require some substantial prior knowledge about the given task. In this work we propose a deep algorithm which learns a deep representation of the environment, its current state and the occurred events to automatically shape the reward function based on the agent's curiosity and current performance.

The method we introduce in this paper, besides the regular reward from the environment, additionally rewards the RL agent by the novelty (or rarity) of the experienced events (or transitions, which are defined by the occurred in-game events and general observations about the environment which are acquired via game screen). The idea is to observe the events, behaviors and states the agent uses and visits, and incite it to explore new states and behaviors. So, we make the agent balance between the rewards which it receives from the environment and curiosity about yet unknown and risky, but potentially rewarding strategies.

AISMA-2021: International Workshop on Advanced in Information Security Management and Applications, Stavropol, Krasnoyarsk, Russia, October 1, 2021

kerrovitarr@gmail.com (V. Sopov); iamakarov@hse.ru (I. Makarov)
0000-0002-3308-8825 (I. Makarov)

CEUR Workshop Proceedings (CEUR-WS.org)

While the method is general enough to be used for any environment and applied to any reward-based RL algorithm, in this paper we study the application of our method to A2C algorithm in VizDoom environment.

2. Related Work

2.1. Deep Reinforcement Learning

Reinforcement learning, while studied and analyzed since 20th century [1] , recently due to the rapid advancements in hardware and development of neural networks have gained a lot more traction. Incorporation of some abilities of neural networks into existing RL algorithms such as Q-learning [2] allowed new algorithms to estimate and model considerably more complex environments such as Atari games [2], shooters such as Doom and even modern games such as Dota 2 [3] and Starcraft 2 [4]. The main goal for these algorithms is to estimate game state and conduct agent strategies based only on the observable space which is the game screen seen by human players. Convolutional neural networks are used by most agents to build powerful embeddings of the screen space.

2.2. Actor-Critic methods

One of the most numerous families of algorithms in reinforcement learning is actor-critic algorithms [5]. Basically such agents have two parts, one of which predicts the action that the agent must take and the other estimates the value of the action that the agent made. Such methods combine simplicity of direct policy optimization such as Policy Gradient and robustness of value-estimating methods such as Q-learning.

A way to greatly enhance the performance of such methods is to train multiple agents simultaneously and asynchronously, collecting agents' transitions and periodically updating the network weights with them. It leads to much faster training and considerably better final performance. This method is called A3C (Asynchronous Advantage Actor-Critic) [6].

At the same time, some researchers consider that the asynchronous algorithm does not in fact contribute to the performance of the algorithm. OpenAI implementation [7] gets rid of it by applying synchronous network updates and making agents' steps synchronously. Their evaluation shows that such method is both more cost-effective and results in better agent performance. The resulting algorithm is called A2C. It will be used as a base RL algorithm in this work and also as a baseline during evaluation.

2.3. VizDoom Environment

It is important to have proper simulation for complex first-person shooters [8, 9, 10, 11, 12, 13] or 2D interaction games [14, 15, 16]. VizDoom [17] as an AI research platform based on an opensource implementation of the original Doom game. It provides API for running games, acquiring both visual frame information and internal game variables, controlling the ingame agent in a flexible and functional way. It also defines the means to create custom scenarios and also provides a set of default scenarios of big variety. They all differ in difficulty, some of them featuring sparse and delayed rewards.

The code of this work is based on an opensource implementation of OpenAI's A2C algorithm made by p-kar [18]. The algorithm was already modified for usage with VizDoom environment and thus is used as the baseline and a starting point for our own work.

2.4. Curiosity, Intrinsic Motivation and Novelty Search

Recent advancements in the field were made partially due to the usage of intrinsic motivation by rewarding agent for implementing new behavior and reaching new states. Making agents curious and more keen to explore new behaviors and states considerably improved performance in many environments, in particular the ones with sparse and delayed rewards [19, 20, 21, 22].

One of the approaches to modelling curiosity is to approximate the frequency of visiting specific states. While for complex environments such counts are close to zero in most cases and are also computationally hard to store and compute, attempts were made to make an approximation of the state density function called pseudocount [23] and then to simplify and approximate it by usage of neural networks [24].

Another approach is to acquire some additional information from the game by the usage of game variables such as player's position, number of kills, number of shots, acquired weapons, etc. The method manually defines a set of events based on these variables, stores history of recent events and compares them with the current moment to approximate the novelty of the state the agent is in and additionally rewards the agent, thus motivating it to seek new states and try new behaviours. The proposed approach is called Rarity of Events (RoE) [25]. This work heavily relies on VizDoom environment for experiments and thus will be one of the main methods to compare our approach with.

3. Data

The tasks considered in this work will be some wellknown scenarios for VizDoom: Health Gathering Supreme, My Way Home, Deadly Corridor, Deathmatch. All of them are characterized by sparse rewards.

Health Gathering Supreme is a scenario where a player find himself in a maze filled with healthkits. The agent must collect this kits, otherwise he will die, because of the poisonous floor which deals damage every second. The goal is to survive as long as possible.

My Way Home is a scenario where the agent must find its way to the armor set located somewhere in the maze. Each second the agent loses some points, so the faster it finds the armor, the more points it will save. Agent will only gain positive reward when he gets to the armor set.

Deadly Corridor is a scenario where the agent must pass through the straight corridor with enemies spawning on the sides. To pass the corridor the agent must firstly defeat all enemies, otherwise it will be impossible to reach the target.

Deathmatch is a complex map which consists of the square main area and four rooms filled with healthkits, armor, ammo and various weapons. Various types of enemies periodically spawn at random points of the main area. The agent gets points by defeating enemies. The goal is to get as many kills as possible before the agent is killed.

As a representation of each state the greyscale 84x84 image of the screen is used. The user interface and the croosshair are not rendered. The display of the player's weapon is turned on for Deathmatch scenario and turned off for all the others. For each game frame we also have access for some basic game variables such as player's coordinates, selected weapon and kill count for each weapon which are then used for computation of the reward function. These variables are not directly used by the RL agent.

4. Proposed Approach

In this work we do not aim to enhance, tune or in any way modify A2C algorithm which will be used in our experiments. Instead our goal is to enhance the environment's reward function in a robust and general way, so that with minimal human supervision (which basically consists of hyperparameter tuning) this enhancement alone would allow the agent to train faster and converge to better strategies which were initially unreachable.

In this work we try to predict the novelty of the occurred transition. By the transition here we mean the occurrence of manually-defined events (such as player movement, kills by any weapon, etc.) combined in either the current frame (its visual representation) or the difference between the current and the previous frames (both alternatives are considered and later tested). If we define some gamescreen state as S, agent's action as a, new state which was reached after the action as S^0, and a set of in-game events occurred as a result of the action as $E = \{move, shoot, kill\}$ (as an example), then the previously mentioned transition will be stored as (S, S^0, E). To predict the novelty, we maintain a history buffer of some length of all recent transitions (sampled with some probability).

As was previously mentioned, existing ways to estimate the transition novelty included estimation of state density functions by the usage of neural networks, implementing pseudocount and deep pseudocount functions to count the number of visits for each state, counting occurred events. In this work we try to estimate the transition novelty by simply training an binary classifier which predicts the probability that a given transition occurred in some recent past. We intentionally limit the maximal age of the recorded and stored transitions to view the ones which occurred only somewhere in the past as novel.

The main challenge in this approach is data acquisition. It is easy to get positive samples (when a transition has occurred), but not quite clear how to get not yet visited transitions. Viewing a part of the acquired data as unvisited means that the classifier will be trained to classify many actually seen transitions as unseen which is intentionally misleading. Getting negative samples either outside of the training process or generating it from some random distribution with some degree of reliability and similarity to real data does not seem clear or feasible.

So, we split the acquired transition data into several buffers and train a separate classifier using its corresponding buffer as positive samples and all other buffers as negative samples. So, each classifier is trained to distinguish its own transitions from the transitions in other buffers. The main assumption here is that if a transition has occurred recently, than it is contained in at least one buffer, and at least one classifier (ideally exactly one) classifies this transition as visited. But if the transition has never occurred, then all classifiers ideally view it as unvisited. More formally, if we train N classifiers and for some transition tr we estimate the probabilties that each classifier has seen the transition as $p_1(tr), p_2(tr), \ldots, p_N(tr)$, then we estimate the transition novelty as $Nov(tr) = 1 - \max_i p_i(tr)$.

The initial idea was to train N identical novelty networks, where we maintain N separate non-intersecting history buffers and train each network to discriminate between its own transitions and transitions of all the other networks by using binary cross-entropy loss. The ”curiosity bonus” given will be than defined as $R^0(tr) = \alpha Nov(tr)$, where α is a parameter tuned to make the bonus comparable to environment reward. The new reward function would then be a sum of the reward provided by the environment and our bonus.

However, the general performance and training speed in this case were unsatisfying. So instead of N networks, a single network with N heads is trained. When the last layer of each network originally outputted a single value, now it outputs an N-dimensional vector, where N equals to the number of history buffers we maintain. It allows to both better incorporate the information from all buffers into a single network to generate a better embedding of the game state, and also lower training and inference time, since generally computations for most layers are be made only once instead of N times.

4.1. A2C Agent

In this work we use A2C (synchronous modification of A3C, Asynchronous Advantage Actor-Critic) as a reinforcement learning algorithm to add our reward function to and also as a baseline for comparison purposes.

The agent’s neural network consists of 3 convolutional layers, followed by two dense layers. For the convolutional layers, their kernels are 8×8, 4×4 and 3×3, their strides are 4, 2 and 1 and the numbers of produced feature maps are 32, 64 and 32. The composition of the convolutional layers produces 32 feature maps of size 7×7. After that goes the common fully-connected layer which flattened input’s size is 1568 units and which output size is 512 units. This common layer has two dense descendants: one corresponds to the Actor part and outputs a softmax-adjusted vector which size is the number of possible actions for the agent. This number differs for each scenario. The second descendant corresponds to the Critic part and outputs a one-dimensional vector.

4.2. Novelty Network

The network for estimating a reward function consists of the convolutional part and the fully-connected part. The convolutional part is the same as in the A2C agent. The output of this part is then flattened and concatenated to the events vector.

Events vector is the vector which corresponds to the events which occured either in the current frame or in the current episode up to this frame (we describe this later when we list all modifications of the proposed algorithm). Its elements are repeated several times (currently 15 times) to have more influence in the resulted vector.

The resulting vector is then fed into a sequence of two fully connected layers that output 512-

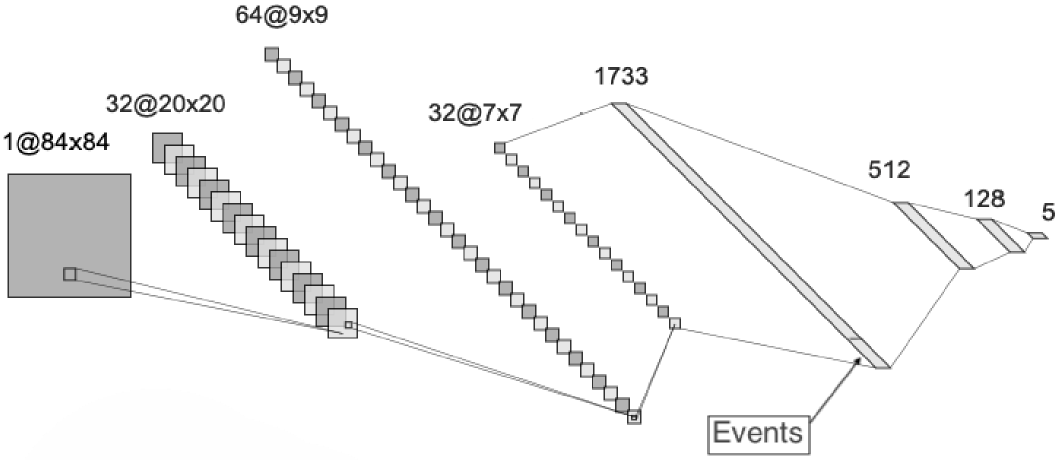

Figure 1: Architecture of Novelty prediction network

and 128-dimensional vectors correspondingly. The final layer is also fully-connected and outputs N-dimentional vector, where N equals to the amount of ”heads”, or ”discriminators”. Sigmoid activation function is then applied to each head independently. The visual representation of novel network can be found in Figure 1.

5. Modifications

Our approach is represented in several modifications, most of which can be applied independently from each other. While the basic algorithm uses both screen info and events data, these information channels can be disabled separately to evaluate their importance, so that the novelty network could be trained only on screen buffer or only on game events.

The number of heads, while could be customized, is set to 5 which turned out to show the best performance. The history buffer size is another numerical parameter which could be tuned. In most cases it is set to 5000 samples per each of 5 heads and is generally bounded by the size of GPU memory and training speed considerations.

A more qualitative modification is the way to feed the game screen into the novelty network. While originally the unmodified image is used, alternatively we could store an element-wise difference between the current state’s matrix and the matrix of the previously observed state. While the former alternative leads to the estimation of the novelty of the current state, the latter focuses more on the dynamics and represents an actual transition transition which is tightly bound to the occurred event. We will call this modification *statediff*.

Another important modification is the cumulativity of the events. Originally, each event is taken only at the exact moment and does not include observations from the previous frames of the episode. As an alternative, another approach is implemented, where each event fed into the novelty network is in fact a normalized cumulative sum among all events which happened until current moment in the given episode:

$$Event_k^{cum} = \frac{1}{k} \sum_i Event_i.$$

The idea behind this is that while ideally the present events are a better indicator of the current state, their accumulation shows how the current state was reached and thus contributes to a better state representation. We will call this modification *episodic*.

Other modifications mainly involve events, their episodic normalization and acquirement. For instance, the accumulated episodic events can be normalized either by the amount of frames or the amount of actual events which contributed to the accumulated sum:

$$Event_k^{cum} = \frac{1}{N_k} \sum_i Event_i,$$

$$N_k = \mid \{Event_i : i \leq k, \| Event_i \|_1 > 0\} \mid$$

While the former alternative shows a better representation of the events density during the game episode, the latter allows us to focus on important contributing events and learn to differentiate between them. Because this is a fix of the averaging in a way, we will call it *fixedaverage*.

Also, some events could be considered too frequent to contribute to the episodic sum (such as player's movement, which is empirically the most frequent event and thus the least meaningful), thus in another modification of the algorithm they can be ignored and not stored in history buffers.

Finally, in another modification it is tested whether the lesser amount of non-contributing (empty, zero) events in the history buffer lead to better performance. While in the basic algorithm all events are acquired with constant probability $p_{base} = 0.1$, for empty events an additional penalty p_{zero}, is introduced, so for such events their probability to be stored in the buffer becomes $p_0 = p_{base} \cdot p_{zero}$. Such modifications will be denoted with *zeroproba*.

6. Training

Conceptually the main training loop of the original A2C algorithm is not changed. The addition we make is that after each A2C model update we also feed all history buffers to the novelty network and compute binary cross-entropy loss for each head. The losses are then summed and minimized using the same optimizer as for the original A2C network, which is RMSProp.

The duration of training is measured in game frames which are played by the agent. Since we use A2C and actually simultaneously have 16 agents, we sum up all the frames of all agents. The number of training frames for each scenario varies. For "My Way Home" we train agents for 4 million frames, For "Health Gathering" scenarios the agents are trained for 3 million frames. For Deathmatch we use 14 million frames for comparison of different modifications of our algorithm and 25 million frames for the comparison of our best modification with the baseline [18] and RoE algorithm [25].

One important note here is that for some scenarios these numbers are actually not enough to fully converge to a best possible strategy for the agent. For example, in the previously mentioned RoE method [25] for the ”Deathmatch” scenario the agent was trained for 70 million frames. For other scenarios the agent was trained for 10 million franes. Unfortunately, for our budget such durations would be unaffordable both in terms of time and money, so we had to lower this duration for the results to be both affordable and still useful. Another concession we had to make was the number of experiments. Most experiment were run once or twice, so no recorded information on the variance could be given. On the other hand, generally the conceptual results tended to coincide among all runs, so our results are good enough to compare different algorithms and their modifications with good degree of reliability.

7. Metrics

Main metrics for evaluation are mean reward and maximal reward. Periodically (after every Nth network update) for every worker (out of 16) the reward of the last finished episode is taken. After that either the maximal value or the mean value among all workers is computed. Since the values are very noisy, for the evaluation and plotting exponential smoothing with $weight = 0.95$ is applied.

8. Experiment Evaluation

Since we are evaluating a reinforcement learning algorithm, no specific evaluation separate from the training process was applied, and rewards acquired during training were evaluated. Mainly the maximal values and their speed of growth are considered.

Also, most experiments are applied to ”Deathmatch” scenario, and the algorithm with the best parameters is later evaluated on other scenarios such as ”My Way Home”, ”Health Gathering Supreme” and ”Deadly Corridor”. The only modified parameter for these scenarios is the bonus coefficient, which determines with which weight the novelty bonus is added to the environmental reward and which is is manually tuned to be approximately equal to the reward at the beginning of the agent’s training.

Table 1
Deathmatch best results

Modifications	Mean Score
Rarity of Events	14.82
episodic, fixedaverage, statediff	12.04
No events	11.08
No modifications	10.74
episodic, fixedaverage, no movement events, zeroproba=0.1	10.31
A2C	9.00

9. Results

In Deathmatch scenario in short-term some modifications manage to beat the baseline A2C algorithm, but do not get better performance than the Rarity of Events method. Among all modifications the best one uses episodically accumulated events which are normalized among non-empty elements and uses difference between current screen state and the previous observation. This modification is used for more lengthy training session on Deathmatch and for all other scenarios. The results of the Deathmatch experiments are shown in Table 1 and Fig 2. The result of the longer training session on Deathmatch is shown in Figure 3.

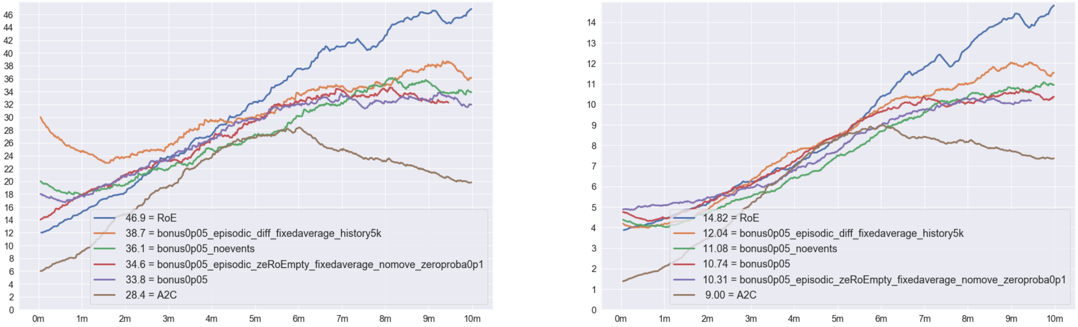

Figure 2: Different variations of the algorithm, Deathmatch, Mean (left) and Max (right) rewards, respectively

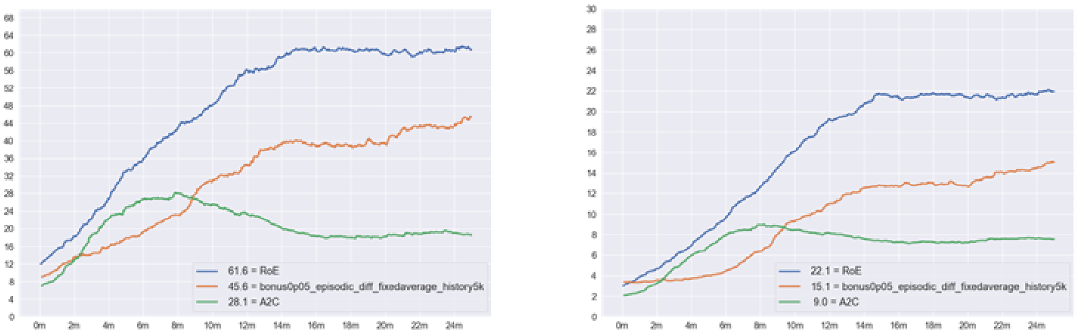

Figure 3: Comparison with existing algorithms, Deathmatch, Mean (left) and Max (right) rewards, respectively

While experiments on Deathmatch scenario are not very satisfying, results on other scenarios show much more promise. In a "My Way Home" scenario, while the baseline A2C agent learns well and quickly enough to make the usage of additional intrinsic rewards unneeded, our methods still leads to faster learning, even if in this case it is not strictly necessary. At the same time, RoE algorithm for some reason shows considerable decline in the performance, most likely connected to the absence of extrinsic reward. The results are shown in Figure 4.

For "Health Gathering Supreme" Scenario, while at the beginning being only marginally better than the baseline and considerably worse than RoE method, our algorithm at the end of

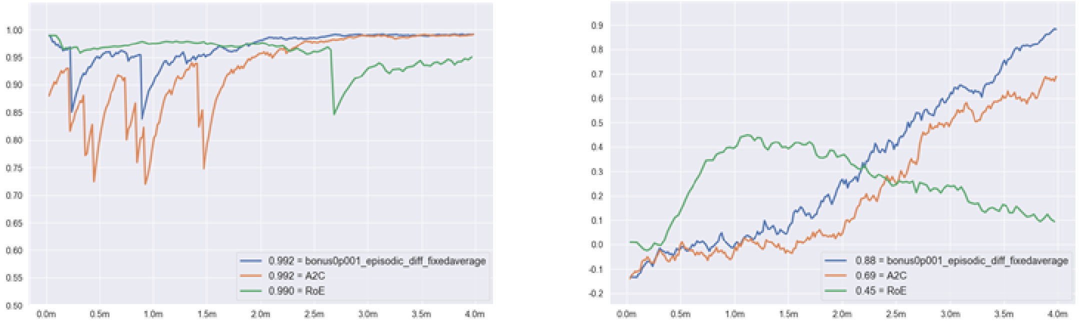

Figure 4: Comparison with existing algorithms, My Way Home, Mean (left) and Max (right) rewards, respectively

the training session becomes the best. And while the advantage appears only at the and and is not significant, the dynamics of the mean reward for RoE method shows that its further rise is unlikely, so the gap will probably only rise. The results are shown in Figure 5.

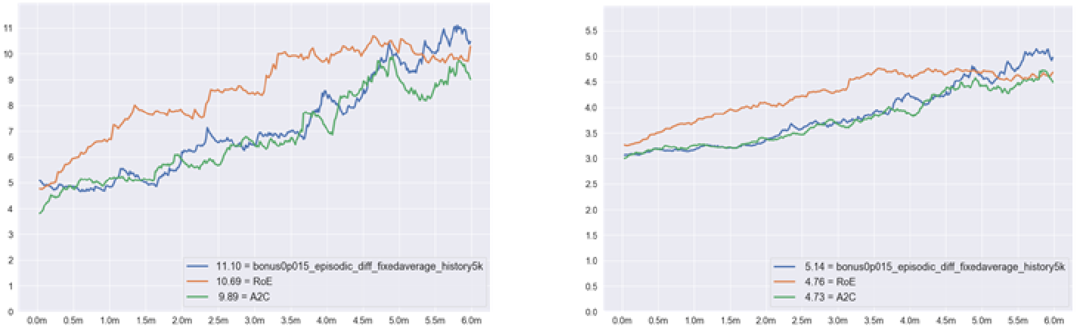

Figure 5: Comparison with existing algorithms, Health Gathering Supreme, Mean (left) and Max (right) rewards, respectively

In ”Deadly Corridor” our method shows the best results. In only 1 million frames it converges to the maximal reward for this scenario and continues to steadily raise its mean reward, while A2C shows stably bad scores. RoE method here has the maximal reward somewhere in the middle of both other methods, while at the same time being only marginally better than A2C baseline for its mean reward without any positive trends. The results are shown in Figure 6.

10. Discussion

Our approach performs reasonably well in several challenging VizDoom scenarios. Still, it should be tested in other scenarios and in conjunction with other reinforcement learning algorithms to test it generality and usefulness. Besides, a more thorough testing should be done, with longer training sessions and with overall more runs per each setup to have a better understanding of the dispersion and general stability of results as well as better understanding of our algorithm’s behavior on later stages of training (this however requires significantly larger

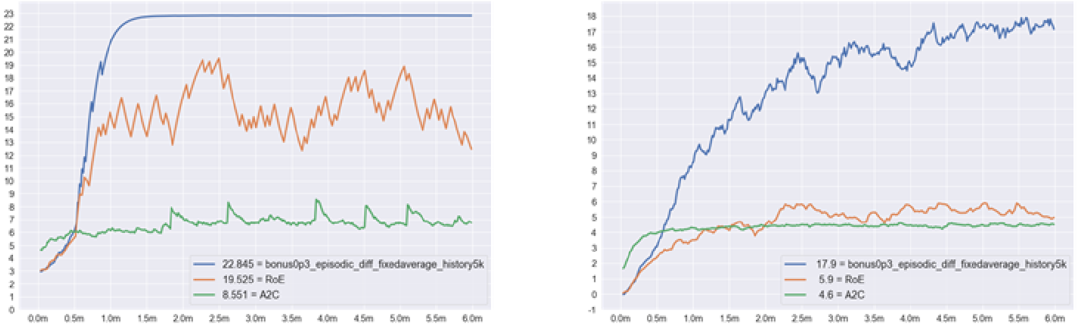

Figure 6: Comparison with existing algorithms, Deadly Corridor, Mean (left) and Max (right) rewards, respectively

funding).

Another point to explore would be the normalization factor which we used to equalize environment's extrinsic reward and our intrinsic bonus. In this work this coefficient was manually tuned for each scenario to make the bonus roughly comparable to the reward. However, no thorough optimization took place to determine best possible values. This could be explored in later works as well as automatic search of the best value of this coefficient.

In parallel, a related topic could be researched. In RoE [25] algorithm extrinsic reward was completely ignored, which resulted in some interesting new behavior patterns and overall significant improvements over the baseline. Since our work works in the same field and utilizes the same concepts, it would be an interesting area for future research.

Speaking of general performance, our method showed some marginal improvements in simple scenarios such as ”My Way Home”, which were not very challenging for the chosen baseline. The improvements in more complex scenarios such as ”Health Gathering Supreme” and ”Deadly Corridor” were much more notable. Howerever, for arguably the most complex scenario ”Deathmatch” our approach, while being better than the baseline, was significantly worse than RoE method. An interesting hypothesis to test is whether it is because of the insufficient quality of our novelty estimation network. While the general concept showed its potential, further research is needed to determine the best neural architecture and further enhance its learning process.

11. Conclusion

In this work we introduced a novelty-based reinforcement learning approach, which focused on the reward function and incited the RL agent to explore new behaviors. Our method in conjunction with A2C algorithm was able to achieve significantly better performance than the baseline A2C algorithm and comparable (or sometimes even better) performance to another novelty-based algorithm RoE. In future, the proposed method could be incorporated into other reward-based reinforcement learning algorithms and reused in other challenging environments with sparse or delayed rewards for which converging to satisfying strategies without any additional intrinsic motivation is unfeasible.

References

[1] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.

[2] 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 (2015) 529–533.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al., Dota 2 with large scale deep reinforcement learning, arXiv preprint arXiv:1912.06680 (2019).

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., Grandmaster level in starcraft ii using multi-agent reinforcement learning, Nature 575 (2019) 350–354.

[5] V. R. Konda, J. N. Tsitsiklis, Onactor-critic algorithms, SIAM journal on Control and Optimization 42 (2003) 1143–1166.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, PMLR, 2016, pp. 1928–1937.

[7] Y. Wu, E. Mansimov, S. Liao, A. Radford, J. Schulman, Openai baselines: Acktr & a2c, url: https://openai. com/blog/baselines-acktr-a2c (2017).

[8] I. Makarov et al., First-person shooter game for virtual reality headset with advanced multi-agent intelligent system, in: Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 735–736.

[9] I. Makarov, M. Tokmakov, L. Tokmakova, Imitation of human behavior in 3d-shooter game, AIST'2015 Analysis of Images, Social Networks and Texts (2015) 64.

[10] I. Makarov et al., Modelling human-like behavior through reward-based approach in a first-person shooter game, in: EEML Proceedings, 2016, pp. 24–33.

[11] I. Makarov, P. Polyakov, R. Karpichev, Voronoi-based path planning based on visibility and kill/death ratio tactical component, in: Proceedings of AIST, 2018, pp. 129–140.

[12] I. Makarov, P. Polyakov, Smoothing voronoi-based path with minimized length and visibility using composite bezier curves, in: AIST (Supplement), 2016, pp. 191–202.

[13] I. Makarov, O. Konoplia, P. Polyakov, M. Martynov, P. Zyuzin, O. Gerasimova, V. Bodishtianu, Adapting first-person shooter video game for playing with virtual reality headsets., in: FLAIRS Conference, 2017, pp. 412–416.

[14] I. Makarov, D. Savostyanov, B. Litvyakov, D. I. Ignatov, Predicting winning team and probabilistic ratings in "dota 2" and "counter-strike: Global offensive" video games, in: Proceedings of AIST, Springer, 2017, pp. 183–196.

[15] I. Kamaldinov, I. Makarov, Deep reinforcement learning in match-3 game, in: Proceedings of CoG, IEEE, 2019, pp. 1–4.

[16] I. Kamaldinov, I. Makarov, Deep reinforcement learning methods in match-3 game, in: Proceedings of AIST, Springer, 2019, pp. 51–62.

[17] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaśkowski, Vizdoom: A doom-based ai research platform for visual reinforcement learning, in: 2016 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2016, pp. 1–8.

[18] P. Kar, A2c, acktr and a2t implementations for vizdoom, 2017. URL: https://github.com/

p-kar/a2c-acktr-vizdoom.
[19] I. Makarov, A. Kashin, A. Korinevskaya, Learning to play pong video game via deep reinforcement learning, in: AIST (Supplement), 2017, pp. 236–241.
[20] D. Akimov, I. Makarov, Deep reinforcement learning with vizdoom first-person shooter, in: CEUR Workshop Proceedings, volume 2479, 2019, pp. 3–17.
[21] M. Bakhanova, I. Makarov, Deep reinforcement learning in vizdoom via dqn and actor-critic agents, in: Proceedings of IWANN, Springer, 2021, pp. 138–150.
[22] A. Zakharenkov, I. Makarov, Deep reinforcement learning with dqn vs. ppo in vizdoom, in: Proceedings of CINTI'21, IEEE, 2021, pp. 1–6.
[23] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, Advances in neural information processing systems 29 (2016) 1471–1479.
[24] G. Ostrovski, M. G. Bellemare, A. Oord, R. Munos, Count-based exploration with neural density models, in: International conference on machine learning, PMLR, 2017, pp. 2721–2730.
[25] N. Justesen, S. Risi, Automated curriculum learning by rewarding temporally rare events, in: 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2018, pp. 1–8.