Segmented Actor-Critic-Advantage Architecture for Reinforcement Learning Tasks

Martin Kaloev, Georgi Krastev

University of Ruse, Department of Computer systems and technologies, Ruse, Bulgaria

Abstract **– The article focuses on experiments with a multi module neural networks type of architecture for neuron-like machine used in reinforcing learning. This type of architecture can be used to solve complex robotic or policy optimization tasks and allows segmented storage of trained memory. Such technique speeds up the training process compared to existing actor-critical algorithms.**

Keywords **– Reinforcement learning, Q-learning, Actor-critic algorithm, Neuron-like machine architecture.**

1. Introduction

Reinforcement learning (RL) refers to a sector in machine learning in which an agent is trained in a reinforcement loop [8], in which each action has a certain reward that helps the agent determine the quality of a strategy or policy of actions. RL have two widely accepted approaches. Those being Q-learning [15], [18] and policy optimization [7], [9], [14], [15], [16] training. These two approaches have different advantages and disadvantages.

DOI: 10.18421/TEM111-27
https://doi.org/10.18421/TEM111-27

Corresponding author: Martin Kaloev,
University of Ruse, Department of Computer systems and technologies, Ruse, Bulgaria
Email: kaloev_92@mail.ru

Received: 04 January 2022.
Revised: 05 February 2022.
Accepted: 11 February 2022.
Published: 28 February 2022.

The main advantage of Q-learning (Q-l) methods is easy implementation and quick adaptation to the task. While the advantage of Policy optimization algorithms is solving problems where the local maximum reward and global maximum reward are in different policies [2], [10], [13]. Example for Global and Local Reward focused policy is shown in Figure 1. The considered task described in the current article is related to the training of a machine to maintain balance and a stable gait. This task is highly complex and requires the agent to select a multi-step policy with different global and local maximums of received reward. Valid tactics and common failed strategies that the agent can develop are discussed. The use of a new type of architecture that combines the advantages of Q-l and PO algorithms is explained and discussed. Example for Q-learning is shown in Figure 2.

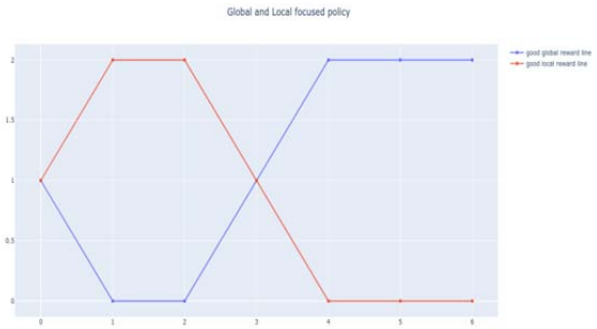

Figure 1. Global and Local Reward focused policy

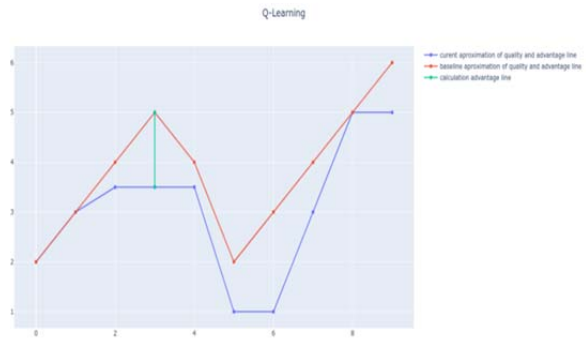

Figure 2. Q-Learning

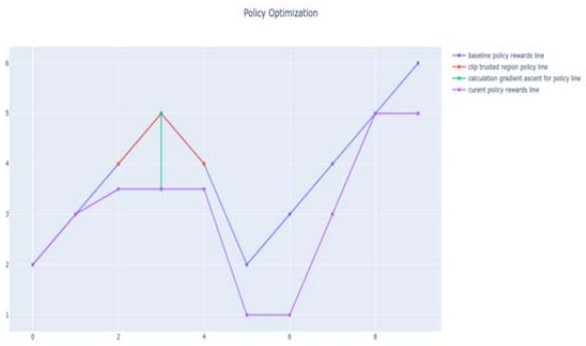

Figure 3. Policy optimization learning

The difference between Q-learning and PO is that one type of algorithm focuses on one point, while the other focuses on a clip of points united in a common policy. Example for Policy optimization learning is shown in Figure 3.

2. Training Task

The used task name is BipedalWalker-v3 [3], [4], a simulation of a bipedal robot that is trained to walk in a rugged area. This type of task has the following characteristics (see Table 1 for action space and see Table 2 for observation state space). As for each successful advance, the agent is given a small reward, and for each loss of balance, the reward is deducted.

Table 1. Action Space

0	Hip_1 (Torque / Velocity) -1 +1
1	Knee_1 (Torque / Velocity) -1 +1
2	Hip_2 (Torque / Velocity) -1 +1
3	Knee_2 (Torque / Velocity) -1 +1

Table 2. Observation State Space

Num	Observation	Factors
0-3	Torso	4
4-13	Legs	8
14-23	Lidar reads	11

3. Tested Architectures

3 types architectures are used for tests. The architectures used for the control tests are A2C [12] and PPO [16]. SA4C is an architecture designed taking into account the results of control tests. This type of architectures has the following elements: an agent that interacts with the environment. An actor's neural network that guides the choice of action and a critique's neural network that calculates the possible benefits of action choice. Example of Q-learning architecture is shown in Figure 4 and for Policy optimization (PO) architecture in Figure 5.

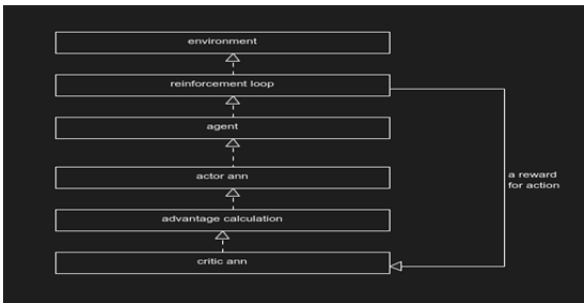

Figure 4. Q-Learning Architecture

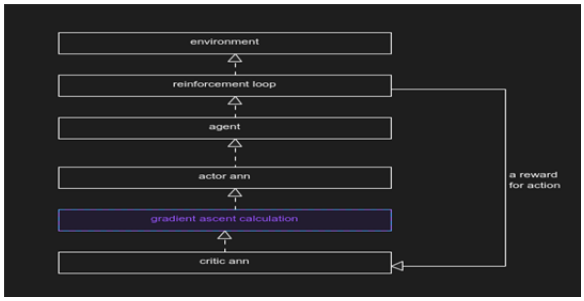

Figure 5. PO Architecture

There is also a variant of the A2C architecture that uses multiple threads to solve the same problem in separate simulations, called A3C [12].

4. Results of Q-l and PO Architectures

The test results show several patterns of successful strategies and several common unsuccessful.

Successful strategies are:

1. Rear balancing, see Figure 6.
2. Forward balancing, see Figure 7.
3. Balancing on knees, see Figure 8.

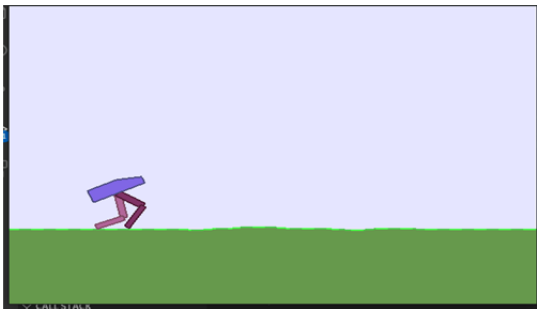

Figure 6. Rear Balancing

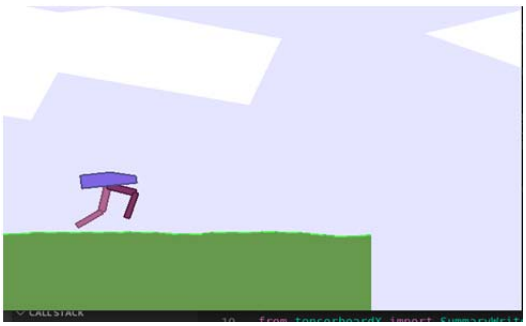

Figure 7. Forward Balancing

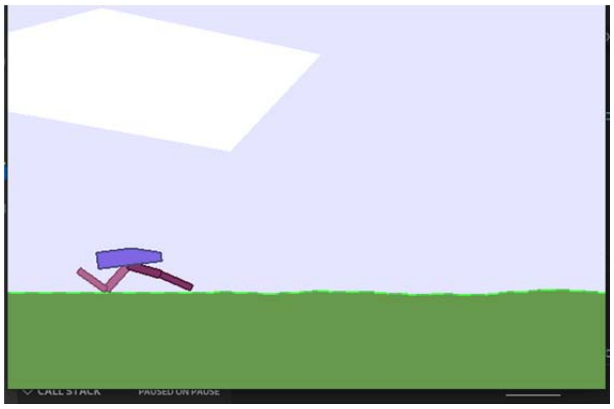

Figure 8. Balancing on Knees

Common failed strategies are:

1. Backflip Jump
2. Freezing
3. Jump Forward
4. Locking

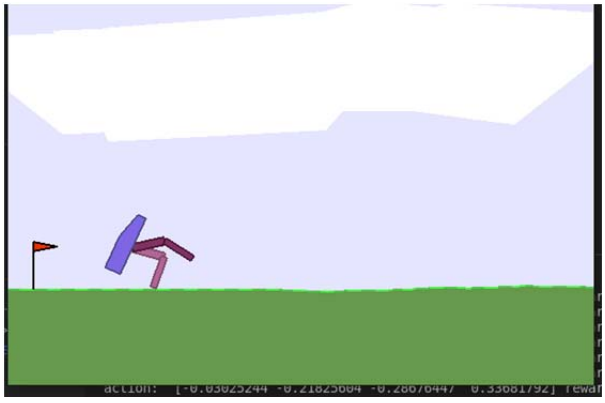

Figure 9. Backflip Jump

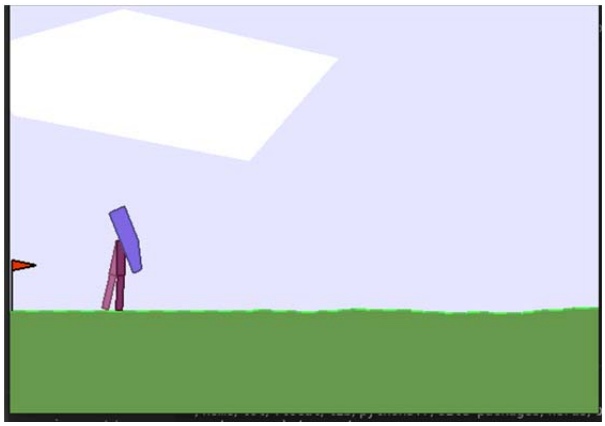

Figure 10. Freezing Type A

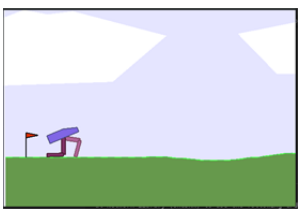

Figure 11. Freezing Type B-1

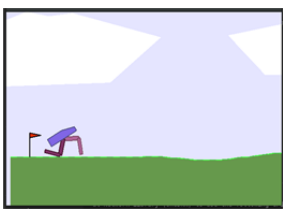

Figure 12. Freezing Type B-2

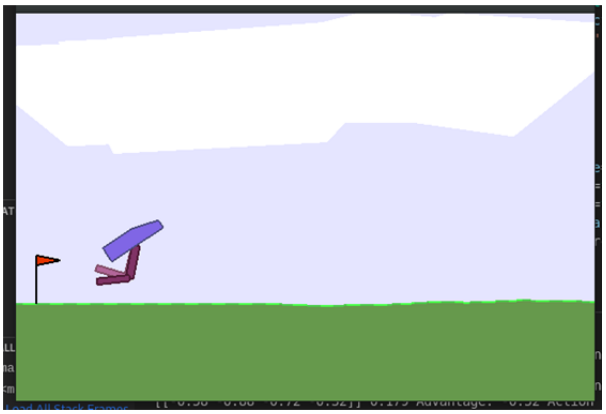

Figure 13. Jump Forward

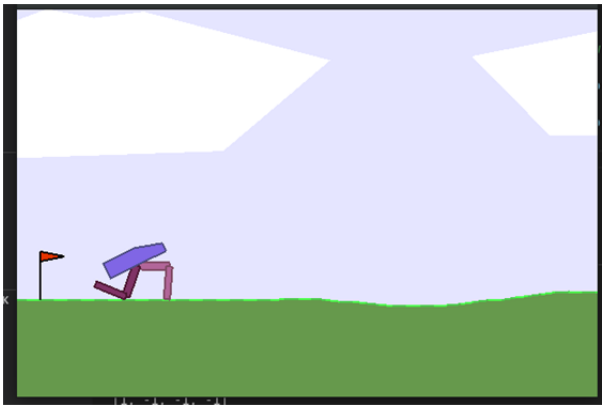

Figure 14. Locking in Bad Profile

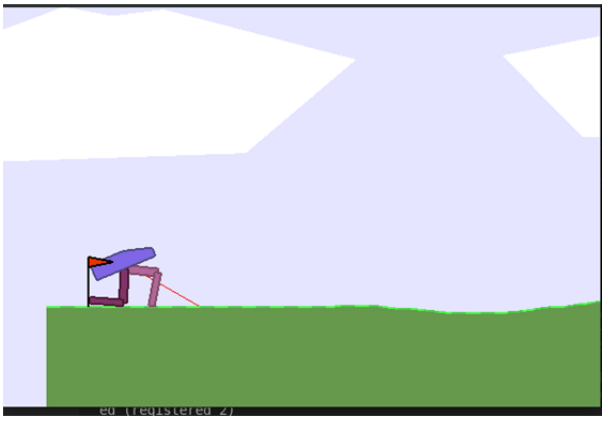

Figure 15. Walking Backwards

5. Explanation of "Locking"

An observed incorrect policy that the agent often learns can be described as locking. This policy is expressed by choosing a uniform combination of actions that the agent assesses as the most correct for each possible combination of factors and conditions. An example of this type of policy can be seen on "Locking in Bad profile", shown in Figure 14. The explanation for this effect is the overfitting of the neural networks governing the actor and critic. When reward-action information is used linearly to train neural networks, each new training example is too similar to the previous one to bring out a noticeably

differences. This means that neural networks have tendency for overestimating the value of actions. This effect can be called overestimation due lacking information caused from overfitting [11]. Wrong prioritization of suboptimal action by an agent in the A2C architecture may also relate to the credit assignment problem [1]. There are techniques to solve this problem, the use of shuffled learning examples and the use of double neural network algorithms [5], [6], [17], example of double DQN architecture, shown in Figure 16.

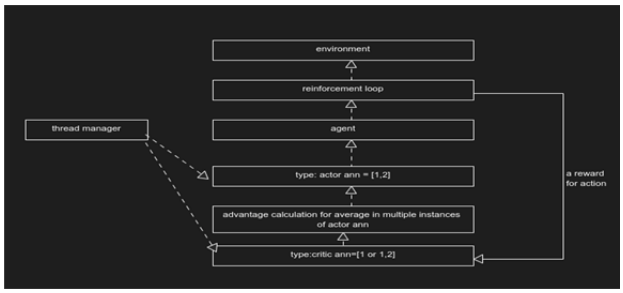

Figure 16. Double DQN

6. Jump Forward

In the training loop of RL related tasks, the agent explores many different policies and strategies. Policies offering the highest possible immediate reward are often examined first. Sometimes this type of policy leads to a closed end and does not allow to reach the global maximum reward. In experiments involving the Bipedal -v3 task, the maximum high immediate reward is reached when the agent makes a long jump at the beginning of the walk. This type of policy has the disadvantage that the agent rarely manages to regain his balance and does not take more than one step. This is an example in which policy sacrifices a global maximum reward for an immediate reward. Example for Jump Forward is shown in Figure 13.

7. Explanation of "Freezing"

In order to avoid the effects described in the previous subsections (5. Locking, 6. Jump Forward), experiments were performed with A2C architecture - double network. The observed results of these experiments can be described as freezing type A) and freezing type B). Freezing type A can be described as freezing of the agent in an equilibrium position, in which the agent chooses not to move forward or backward so as not to risk loss of balance. Example for Freezing type A is shown in Figure 10. This effect is observed in the use of multi-actor architecture and two critics. The two critics are in a cycle that alternates between which to train and which to calculate the existing advantage for action. The agent freezes when the values for the same action are drastically different according to the two critics, which means that the actors are trained that no different action has an advantage over the current action of maintaining balance. Example for calculating benefits that cause conflict of critics can be seen in Figure 17. Freezing type B can be described as a condition in which the agent abruptly loses balance and cannot recover in a cycle of steps. This type of freezing is observed in architecture with many actors and one critic. Note that each neural network is different from other networks. The more different the architectures, the greater the chance of avoiding locking in the same movement. But the downside is that too many different actors reduce the chance of successful coordination in a successful policy. Example of freezing type B, Freezing type B-1 can be seen in Figure 11 and Freezing type B-2 in Figure 12.

Figure 17. Critic ANNs in Conflict

8. Jump Backwards (Backflip jump)

Additional experiments with the reward function show how an agent can learn a policy of jumping back. If an increased penalty is received or the environment is restarted each time the agent loses balance, this forms a behavior in which the agent chooses to make a sharp jump back. Additionally, if the agent receives a negative reward for each forward movement, a behavior is observed in which the agent decides to go in the opposite direction, see Walking Backwards in Figure 15. This shows that it is possible to manipulate the training and behavior of the agent through, subtle changes in the value of the reward. Example for Backflip Jump is shown in Figure 9.

9. Segmented Asymmetric Actor Advantage Critic Architecture (SA4C)

Based on the results obtained and described in the previous subsections, an experiment with a new type of architecture is conducted. It is consisting of multiple actors united in by one critic and one agent for one copy of the environment. This avoids locking or freezing in bad positions. The creation of a trusted region for policy optimization is additionally simulated, but instead of using an algorithmic

approach with gaussian likehood, a mechanical approach is used [9], [14]. Provided that there are a sufficient number of neural networks, it is possible that each individual neural network (actor type) to be responsible for learning and implementing a small step of the overall policy. This allows individual networks to be specialized and to store information step by step. Example for RL in segments is shown in Figure 18. Two successful variants of this type of architecture SA4C with 4 actors are shown in Figure 19 and SA4C with 8 actors in Figure 20 which have been found.

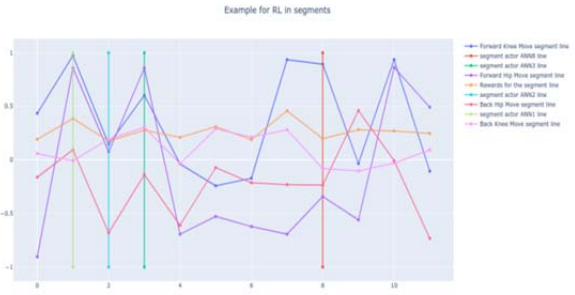

Figure 18. Example for RL in Segments

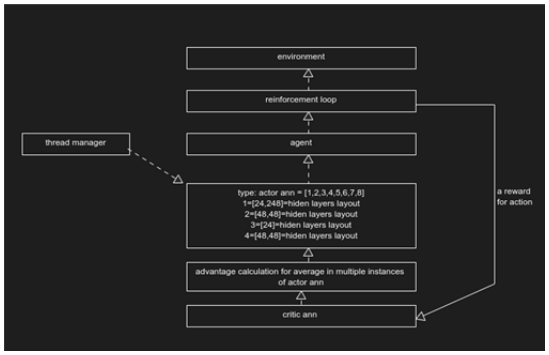

Figure 19. SA4C with 4 Actors ANNs

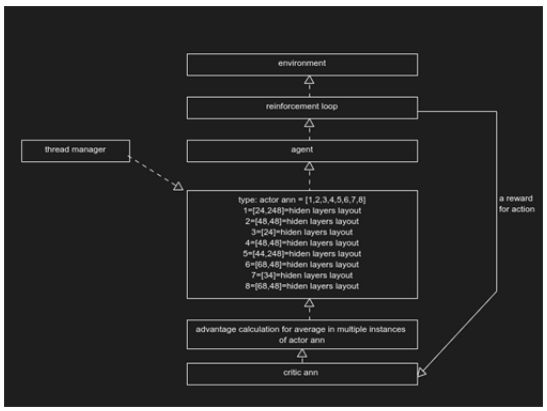

Figure 20. SA4C with 8 Actors ANNs

10. Results from SA4C Tests

Experiments show that the SA4C architecture performs relatively worse than the PPO type architecture. But the main advantage of SA4C is that policy information is segmented. This allows an agent to be trained to handle only a short stretch of policy. Then the network trained by this agent is to be implemented in another architecture. This can avoid exploring bad policies. This allows the introduction of constraints in the research process that are useful in accelerating the solution of the problem. Example of donation of trained ANN is shown in Figure 21. The results can be seen at Rewards per Move for SA4C and PPO Algorithms in Figure 22.

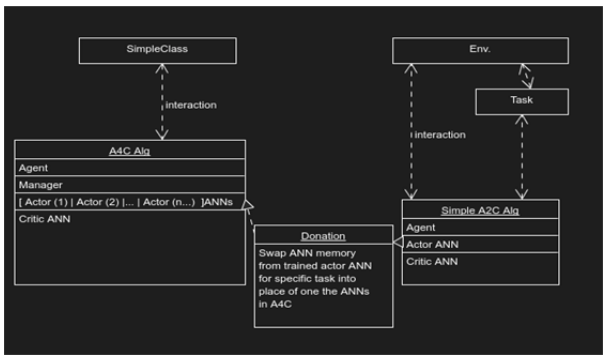

Figure 21. Work flow Chart for Donation of Trained ANN

Example of training agent simple task can be seen in Figure 21 that is part of more complex policy. In this particular example is the first steps robot landing preventing jump forward.

Figure 22. Rewards per Move for SA4C and PPO Algorithms

11. Conclusion

The article discusses an experimental variant of the architecture for neuron-like learning machine with RL. This type of new architecture offers the advantages of the PO-type architecture for solving complex tasks, but it also has the advantages of the Q-type architecture and allows a low number of hidden layers. The advantage of this type of new architecture over existing solutions is the ability to implement tactics in the learning process and thus accelerate the solution of the task. This new option also allows the opportunity to learn and solve problems tailored to specific limitations in the study of the new environment.

Acknowledgement

This publication is developed with the support of Project BG05M2OP001-1.001-0004 UNITe, funded by the Operational Programme "Science and Education for Smart Growth", co-funded by the European Union trough the European Structural and Investment Funds.

References

[1]. Alipov, V., Simmons-Edler, R., Putintsev, N., Kalinin, P., & Vetrov, D. (2021). Towards Practical Credit Assignment for Deep Reinforcement Learning. *arXiv preprint arXiv:2106.04499*.

[2]. Bagnell, J. A., & Ng, A. Y. (2005, December). On local rewards and scaling distributed reinforcement learning. In *Proceedings of the 18th International Conference on Neural Information Processing Systems* (pp. 91-98).

[3]. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[4]. Dibachi, J., & Azoulay, J. (2021). Teaching a Robot to Walk Using Reinforcement Learning. *arXiv preprint arXiv:2112.07031*.

[5]. Fujimoto, S., Hoof, H., & Meger, D. (2018, July). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587-1596). PMLR.

[6]. Hasselt, H. V. (2010, December). Double Q-learning. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 2* (pp. 2613-2621).

[7]. Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., ... & Silver, D. (2017). Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*.

[8]. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237-285.

[9]. Kakade, S., & Langford, J. (2002, July). Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 267-274).

[10]. Mao, H., Gong, Z., & Xiao, Z. (2020). Reward design in cooperative multi-agent reinforcement learning for packet routing. *arXiv preprint arXiv:2003.03433*.

[11]. Meng, L., Gorbet, R., & Kulić, D. (2021, January). The effect of multi-step methods on overestimation in deep reinforcement learning. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 347-353). IEEE.

[12]. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928-1937). PMLR.

[13]. Russell, S. J., & Zimdars, A. (2003). Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (pp. 656-663).

[14]. Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In *International conference on machine learning* (pp. 1889-1897). PMLR.

[15]. Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

[16]. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[17]. Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).

[18]. Watkins, C. J., & Dayan, P. (1992). \cal Q-learning. *Machine learning*, *8*(3-4), 279-292.