УДК 004.853
DOI: 10.18799/29495407/2024/3/61
Шифр специальности ВАК 2.3.1; 2.3.5

Об обучении интеллектуальных агентов в виртуальной среде для задачи управления роботом-манипулятором

Н.Е. Залогин✉, Д.С. Григорьев

Национальный исследовательский Томский политехнический университет, Россия, г. Томск

✉nez4@tpu.ru

Аннотация. ***Актуальность.*** Определяется необходимостью разработки эффективных методов обучения интеллектуальных агентов для задач управления роботами-манипуляторами в виртуальной среде, что критически важно для повышения точности и эффективности промышленных и исследовательских процессов в различных областях. ***Цель.*** Реализация, исследование и модификация алгоритмов обучения с подкреплением, таких как Deep Q-Network (DQN) и Proximal Policy Optimization (PPO), для управления агентами в виртуальной среде KukaDiverseObjectEnv на платформе PyBullet с целью создания моделей, способных точно и надежно взаимодействовать с объектами различных классов. ***Методы.*** Программирование, эксперименты и синтез, сравнительный анализ. ***Результаты и выводы.*** Проведён сравнительный анализ эффективности алгоритмов DQN и PPO, а также модификаций последнего для обучения агентов в конкретной виртуальной среде. Показано, что обученные агенты способны решать поставленную задачу, а модификации позволяют сократить время обучения и количество необходимых шагов в среде. В результате тестирования алгоритмы продемонстрировали приемлемую точность управления манипулятором, что подтверждается на 1000 тестовых эпизодов среды. Реализованные алгоритмы и разработанные модификации обладают потенциалом для использования в промышленных приложениях и дальнейшего развития в реальных условиях, что подчеркивает их значимость для современной робототехники и автоматизации.

Ключевые слова: обучение с подкреплением, виртуальная среда, Deep Q-network, Proximal policy optimization, робот-манипулятор, робототехника

Для цитирования: Залогин Н.Е., Григорьев Д.С. Об обучении интеллектуальных агентов в виртуальной среде для задачи управления роботом-манипулятором // Известия Томского политехнического университета. Промышленная кибернетика. – 2024. – Т. 2. – № 3. – С. 1–8. DOI: 10.18799/29495407/2024/3/61

UDC 004.853
DOI: 10.18799/29495407/2024/3/61

On training intelligent agents in a virtual environment for the robot manipulator control task

N.E. Zalogin✉, D.S. Grigoriev

[1] National Research Tomsk Polytechnic University, Tomsk, Russian Federation

✉nez4@tpu.ru

Abstract. ***Relevance.*** The necessity for developing effective methods for training intelligent agents in robotic manipulator control tasks in a virtual environment is critically important for enhancing the accuracy and efficiency of industrial and research processes across various fields. ***Aim.*** Implementation, investigation and modification of reinforcement learning algorithms, such as Deep Q-Network and Proximal Policy Optimization, to manage agents in the KukaDiverseObjectEnv virtual environment on the PyBullet platform in order to create models that can accurately and reliably interact with objects of different classes. ***Methods.*** Programming, experimentation and synthesis, and comparative analysis. ***Results and conclusions.*** The authors have carried out comparative analysis of the effectiveness of Deep Q-Network and Proximal Policy Optimization algorithms, as well as modifications of the Proximal Policy Optimization for training agents in a particular virtual environ-

ment. It was shown that trained agents are able to solve the assigned task, and modifications can reduce the training time and the number steps required in the environment. As a result of testing, the algorithms demonstrated acceptable accuracy in manipulator control, which is validated on 1000 test episodes of the environment. The implemented algorithms and developed modifications have the potential to be used in industrial applications and further development in real-world conditions, which highlights their importance for modern robotics and automation.

Keywords: reinforcement learning, virtual environment, Deep Q-Network, Proximal Policy Optimization, robot manipulator, robotics

For citation: Zalogin N.E., Grigoriev D.S. On training intelligent agents in a virtual environment for the robot manipulator control task. *Bulletin of the Tomsk Polytechnic University. Industrial Cybernetics*, 2024, vol. 2, no. 3, pp. 1–8. DOI: 10.18799/29495407/2024/3/61

Введение

Внедрение робототехнических решений расширяется на различные сферы деятельности: от промышленных производств до логистики и медицины. Роботы-манипуляторы благодаря своей прецизионности и повторяемости становятся незаменимыми инструментами для выполнения сложных задач.

Одним из основных вызовов в робототехнике является обучение роботов новым навыкам. Традиционные подходы программирования действий роботов требуют временных и трудовых затрат, а также недостаточно эффективны для задач, требующих адаптивности к изменяющимся условиям [1–3].

Тем не менее существуют современные методы решения ряда проблем, основанные на алгоритмах глубокого обучения с подкреплением (*Reinforcement Learning, RL*), которые представляют собой перспективный подход для создания агентов, способных самостоятельно обучаться через метод проб и ошибок и принимать решения в сложных динамических средах, как виртуальных, так и реальных.

Платформы симуляции, такие как PyBullet [4] и Gazebo [5], предоставляют собой настраиваемые среды, используемые для обучения интеллектуальных агентов. Их функционал позволяет оперативно создавать различные сценарии, производить тестирование разработанных алгоритмов, оценивать их эффективность. Несмотря на то, что среды являются неидеальным симулятором (отличаются от реального мира), подобные среды отличает отсутствие проблемы «безопасного обучения», актуальной для реальных роботов.

Применение алгоритмов глубокого RL и их дальнейшее развитие могут привести к созданию более интеллектуальных и адаптивных систем, способных решать сложные задачи и обеспечивать оптимальное поведение в разнообразных условиях, что подтверждается работами [6, 7] исследовательских групп, достигших прогресса в области применения RL для роботов. Основной целью является разработка и тестирование RL-агента в виртуальной среде, а также оценка его эффективности.

Общие подходы и методика исследования

Исследование направлено на изучение методов обучения интеллектуальных агентов на примере использования виртуальной среды KukaDiverseObjectEnv, реализованной на основе PyBullet. Среда позволяет моделировать задачи взаимодействия с объектами, которые требуют от агента высокой точности и координации движений.

Исследования в области RL-агентов, работающих в виртуальных средах с манипуляторами, стремительно развиваются. Одной из ключевых задач в данной области является разработка и исследование алгоритмов, которые могут эффективно обучаться в виртуальных средах, имитирующих реальные задачи манипулирования объектами. Сложность обучения в таких средах заключается в высокой размерности пространства состояний и действий, графических входных данных, а также в необходимости обеспечения стабильности и эффективности обучения агентов. Ряд работ рассматривают применение RL-алгоритмов для решения задач манипулирования объектами, а также множество вопросов, включающих: вопросы безопасности [7], результативности алгоритмов и стратегий для различных сложных виртуальных сред с визуальной информацией [8–11], создания и решения собственных сложных и комплексных сред [5, 6].

Одними из эффективных методов в этой области являются алгоритмы обучения с подкреплением, такие как Deep Q-Network (DQN) [12] и Proximal Policy Optimization (PPO) [13]. Алгоритм DQN использует глубокие нейронные сети для аппроксимации функции награды и позволяет агенту эффективней обучаться оптимальной стратегии. Алгоритм PPO улучшает стабильность обучения за счёт использования доверительных границ для обновления политик, что делает его одним из передовых методов для обучения агентов в условиях непрерывных и дискретных пространств действий.

В данном исследовании основной акцент сделан на имплементацию и улучшение алгоритма PPO. Для уменьшения времени обучения и обеспечения репрезентативности полученных результатов критерием ранней остановки для всех экспериментов

было установлено достижение средней награды в 50 за последние 100 эпизодов, что эквивалентно 50 % точности. Это позволяет существенно сократить время обучения и избежать переобучения, которое часто наблюдается в задачах с высоким уровнем сложности среды.

Для тестирования фактической точности обученных агентов использовалось по 1000 тестовых эпизодов среды. Выбранные критерии оценки позволяют выявить и устранить возможные недостатки в обучении агентов, а также определить, насколько эффективно они могут выполнять поставленную задачу в условиях виртуальной среды.

Платформа симуляции и виртуальная среда обучения

Использованная в данной работе платформа PyBullet, основанная на физическом движке Bullet Physics, представляет собой удобный инструмент для физического моделирования и симуляции роботов. Виртуальная среда KukaDiverseObjectEnv [14], разработанная для обучения агентов управлению роботами-манипуляторами, предоставляет сложную задачу для исследования алгоритмов RL.

Основная задача – научить манипулятор корректно захватывать и поднимать объекты различной сложной формы из контейнера. При каждом шаге в среде манипулятор автоматически опускается по оси z, а агент принимает решение о перемещении по осям x и y и угле поворота захвата. Награда бинарная и выдается, только если один из объектов был захвачен манипулятором и находится выше заданной высоты к концу эпизода. Пространство действий имеет три параметра – по одному действию для перемещения по каждой из осей x и y размерностью от –1,0 до 1,0, а также одно действие для управления углом захвата. Входными данными, предоставляемыми агенту, являются трехканальные изображения состояния среды с виртуальной камеры размерностью (48, 48, 3), пример таких изображений приведен на рис. 1.

Благодаря редкой двоичной награде, графическим входным данным и сложности задачи в целом среда становится достаточно сложной и перспективной для исследования.

Алгоритмы обучения с подкреплением

DQN [12] является *off-policy* [16] алгоритмом и использует нейронную сеть для приближения Q-функции, которая оценивает ожидаемое суммарное дисконтированное вознаграждение агента для каждой пары состояние–действие в среде.

Для тренировки алгоритма DQN будет использоваться воспроизведение опыта (*Experience Replay*). Оно сохраняет последовательности переходов $e_t=(S_t,A_t,R_t,S_{t+1})$ (где e – эпизод обучения, S – множество состояний среды, A – множество действий, R – множество наград), наблюдаемых агентом в реплей буфер $D_t=\{e_1,\ldots,e_t\}$, что позволяет повторно использовать эти данные позже. Путем случайной выборки из памяти достигается декорреляция переходов, из которых формируется пакет данных (Batch). Это значительно стабилизирует и улучшает процедуру обучения алгоритма DQN. Также используется периодическое обновление целевой сети: Q-функция оптимизируется по целевым значениям, обновляемым с определенным периодом. Q-сеть клонируется и «замораживается» как целевая сеть каждые C шагов (C – гиперпараметр). Эта модификация делает процесс обучения более стабильным, так как позволяет избежать краткосрочных колебаний в значениях Q-функции [17].

Необходимо отметить, что входные данные в алгоритме DQN представлены в градациях серого для сокращение затрачиваемых вычислительных ресурсов. Топология нейронной сети DQN представлена на рис. 2.

Рис. 1. *Пример изображений с виртуальной камеры*
Fig. 1. *Example of an images from the virtual camera*

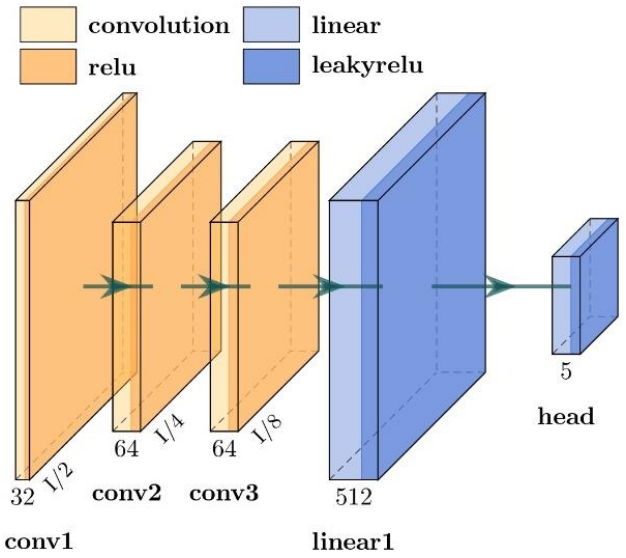


Рис. 2. *Топология нейронной сети агента DQN*
Fig. 2. *Topology of the neural network of the DQN agent*

PPO [13] основан на политике (*on-policy*) [16] и использует *actor-critic* архитектуру [18]. PPO применяет две ключевые идеи для обеспечения стабильности и эффективности обучения. Первая идея – это обновление политики (*policy*) на основе отношения вероятностей между старой и новой политиками, чтобы ограничить ее изменение. Отношение политик вычисляется по формуле (1) [19]. Это означает, что обновление политики происходит малыми шагами и избегает значительных изменений, которые могут приводить к нестабильности обучения. Вторая идея – это использование значений «преимуществ» (*advantages*) для выравнивания обновлений политики, в противовес абсолютным вероятностям. Это способствует более сбалансированному и стабильному обучению.

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}, \quad (1)$$

где π – политика; θ – параметры политики; θ_{old} – вектор параметров политики до обновления; a – действие в среде; s – состояние среды; $\pi(a|s)$ – нейросеть определяющая вероятность выбора действия a в состоянии s.

Без ограничения на расстояния между новой и старой политиками максимизация целевой функции привела бы к нестабильности с чрезвычайно большими обновлениями параметров и большими коэффициентами политики. PPO накладывает ограничение с помощью функции clip(), заставляя значения оставаться в пределах интервала между 1–ε и 1+ε. Вычисляется целевая функция с этими изменениями по формуле (2) [19]:

$$J^{CLIP}(\theta) = E\left[\begin{matrix}\min(r(\theta)\hat{A}_{\theta_{old}}(s,a), \\ clip(r(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_{\theta_{old}}(s,a)\end{matrix}\right], \quad (2)$$

где $A(s,a)$ – функция преимуществ; ε – гиперпараметр.

Кроме ограниченного вознаграждения, целевая функция включает в себя функционал ошибки при оценке значения и энтропией, чтобы стимулировать достаточное исследование среды. В итоге целевая функция будет вычисляться по формуле (3) [19]:

$$J^{CLIP}(\theta) = E\left[\begin{matrix}J^{CLIP}(\theta) - c_1(V_0(s) - V_{target})^2 + \\ +c_2 H(s, \pi_\theta(.))\end{matrix}\right], \quad (3)$$

где c_1 и c_2 – константные гиперпараметры; $V(s)$ – ожидаемый возврат состояния s.

Вместо обновления политики на основе всех данных сразу PPO использует мини-пакеты данных (mini-batch) и SGD (Stochastic Gradient Descent) [20], что позволяет более эффективно применять вычислительные ресурсы и улучшать обобщающую способность модели. Топология нейронной сети PPO представлена на рис. 3.

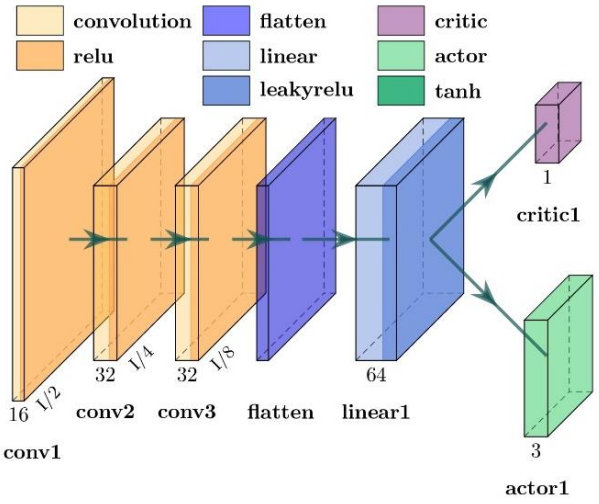


Рис. 3. *Топология нейронной сети агента PPO*
Fig. 3. *Topology of the neural network of the PPO agent*

PPOv1 представляет собой первую модификацию PPO, специально адаптированную для взаимодействия со средой KukaDiverseObjectEnv. В данной модификации, помимо структурных изменений в обучающем цикле, произведены изменения в пространствах наблюдений и действий. Входные данные увеличены с (48, 48, 3) до (84, 84, 3), а количество доступных действий увеличено до пяти: по два действия для перемещения по каждой из

осей *x* и *y*, а также одно действие для управления углом захвата. Изменения также затронули архитектуру нейронной сети. Дополнительные скрытые слои были добавлены для сетей actor-critic. Кроме того, в сверточных слоях изменены размеры ядра и шаг, а также применена пакетная нормализация (BatchNorm2d). В качестве метода инициализации весов нейронной сети использован метод равномерного распределения Ксавье (*xavier-uniform*). Данные изменения должны стабилизировать и ускорить обучение агента. Топология нейронной сети PPOv1 представлена на рис. 4.

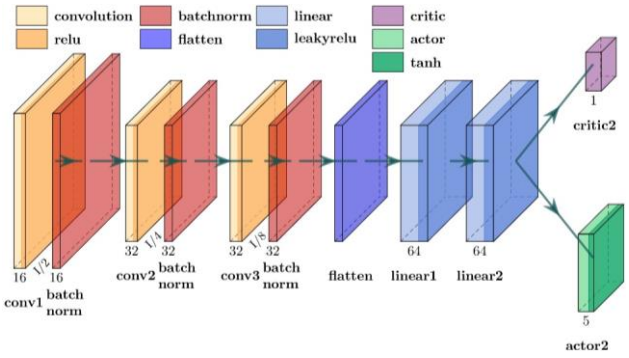


Рис. 4. *Топология нейронной сети агента PPOv1*
Fig. 4. *Topology of the neural network of agent PPOv1*

PPOv2 вносит дополнительные изменения к PPOv1, включая синхронное параллельное выполнение действий в среде. Этот подход существенно ускоряет процесс сбора траекторий действий в среде и, следовательно, улучшает производительность обучения агента в целом. Реализация параллельных сред выполнена с использованием библиотеки «multiprocessing» [21], при этом число параллельных сред составляет 20, что соответствует количеству потоков процессора на рабочей станции.

PPOv3 продолжает развитие PPOv2. В этой версии алгоритма были оптимизированы гиперпараметры благодаря использованию модульной системы обучающего цикла. В частности, главными изменениями стали уменьшение размера пакета данных и увеличение скорости обучения, что позволило сократить длительность эпизодов обучения и ускорить процесс обновления весов модели. Дополнительным изменением является увеличение глубины слоев свертки, включая общие слои (shared layers) и слои actor-critic. Топология нейронной сети PPOv3 представлена на рис. 5.

Алгоритм PPOv4 представляет собой четвертую модификацию PPO. Нововведением данной версии стало добавление механизма внимания (*attention*) [22]. Механизм внимания позволяет модели фокусироваться на различных частях входных данных, взвешивая их значимость для текущей задачи. В контексте робота-манипулятора это означает, что модель должна выделять важные признаки объектов и их окружения, что критически важно для точного выполнения манипуляций. Топология нейронной сети PPOv4 представлена на рис. 6.

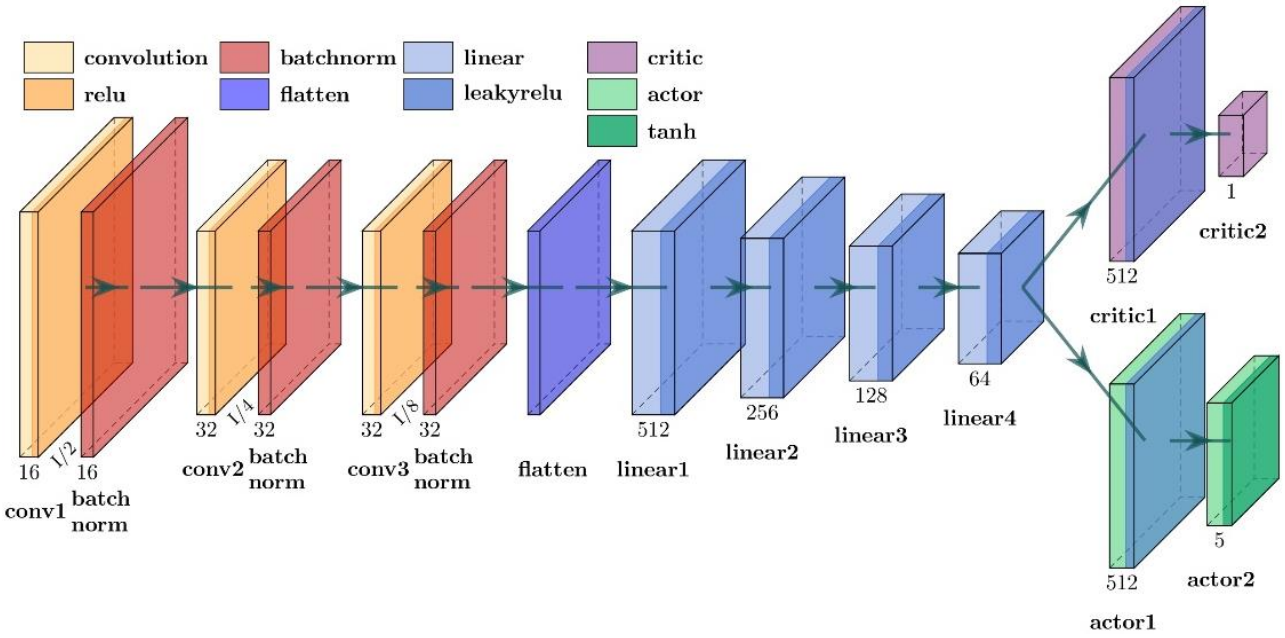


Рис. 5. *Топология нейронной сети агента PPOv3*
Fig. 5. *Topology of the neural network of the PPOv3 agent*

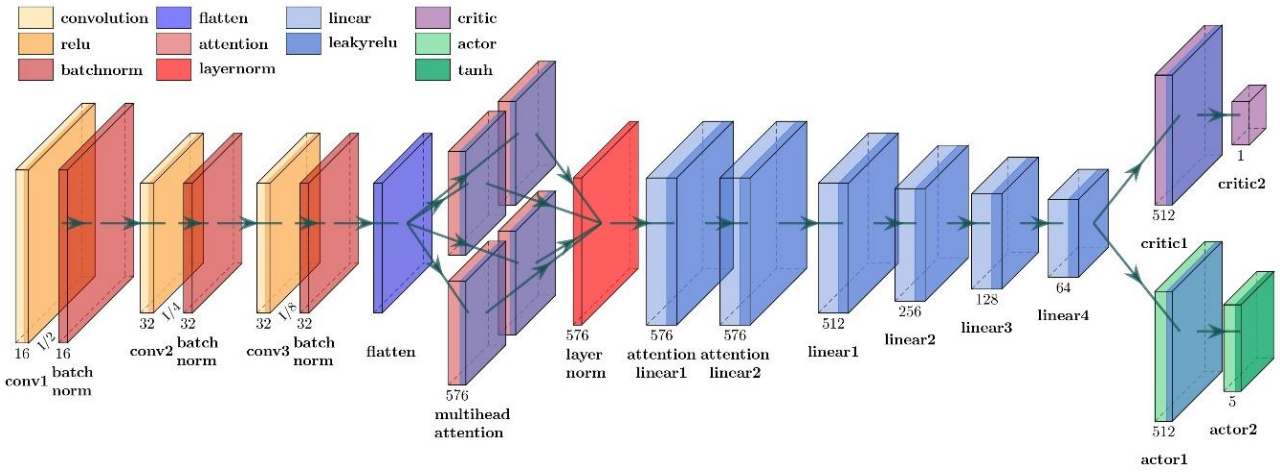


Рис. 6. *Топология нейронной сети агента PPOv4*
Fig. 6. *Topology of the neural network of the PPOv4 agent*

Результаты исследования и их обсуждение

В ходе обучения агентов были проведены измерения средней награды за 100 эпизодов, результаты которых представлены на рис. 7–9.

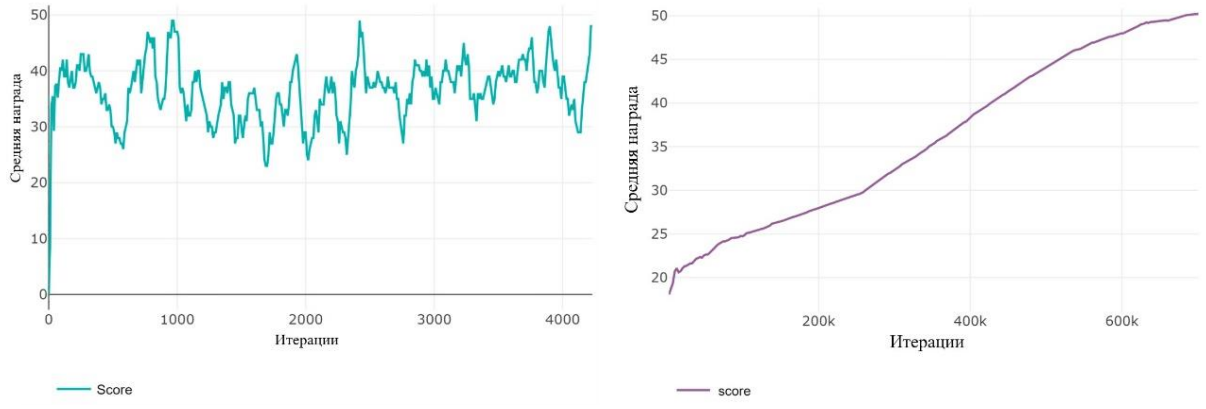
В дополнение к графикам средней награды были зафиксированы показатели конечной средней награды, количество шагов обучения и время обучения, приведенные в табл. 1.

Из результатов, представленных на рис. 7–9 и в табл. 1 можно сделать вывод о том, что стандартные алгоритмы DQN и PPO достигают поставленной цели, но обучение занимает значительное количество времени. Последующие три модификации значительно сокращают время обучения агента и количество шагов в среде. Добавление механизма внимания после сверточных слоев в PPOv4 способствует стабилизации процесса обучения. Тем не менее, несмотря на улучшение стабильности обучения, механизм внимания не оказывает значительного влияния на конечную точность модели. Более того, на начальных этапах обучение модели начинается с существенно более низких значений средней награды, однако впоследствии их производительность выравнивается.

Таблица 1. *Сравнение метрик обучения всех полученных агентов*

Table 1. *Comparison of training metrics of all received agents*

Агент Agent	Средняя награда Average reward	Шаги обучения Training steps	Время обучения (ч:мм:сс) Training time (h:mm:ss)
DQN	51,000	4239	2:06:54,878
PPO	50,202	696320	5:12:04,500
PPOv1	50,690	65 (66560)	1:52:07,842
PPOv2	50,061	26 (26624)	0:35:12,836
PPOv3	50,067	13 (13312)	0:09:34,629
PPOv4	50,229	47 (48128)	0:32:01,375



а/a *б/b*

Рис. 7. *Средняя награда агента DQN (а), средняя награда агента PPO (б)*
Fig. 7. *Average reward of DON agent (a), average reward of PPO agent (b)*

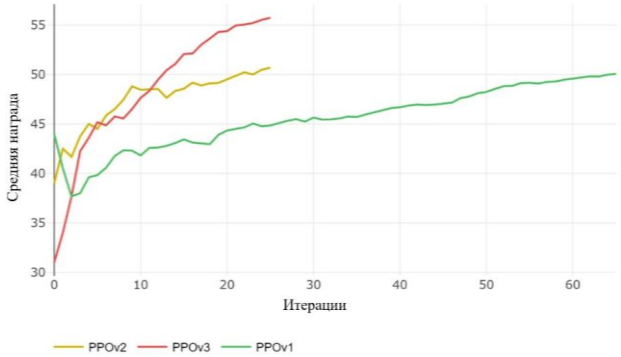


Рис. 8. *Средняя награда агентов PPOv1, PPOv2, PPOv3*
Fig 8. *Average reward of PPOv1, PPOv2, PPOv3 agents*

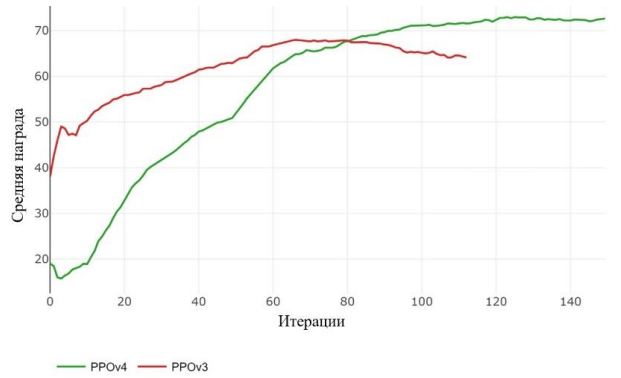


Рис. 9. *Средняя награда агентов PPOv3 и PPOv4*
Fig. 9. *Average reward of PPOv3 and PPOv4 agents*

Для тестирования фактической точности полученных агентов использовалось по 1000 тестовых эпизодов среды. Результаты тестирования приведены в табл. 2.

Как видно из табл. 2, все агенты решают поставленную задачу с заданной точностью в окрестности 50 % за 1000 тестовых эпизодов среды.

Таблица 2. *Тестирование всех обученных агентов*
Table 2. *Testing of all trained agents*

Агент Agent	DQN	PPO	PPOv1	PPOv2	PPOv3	PPOv4
	%					
Положительный эпизод True episode	47,8	49,7	50,2	49,1	52,4	50,7
Отрицательный эпизод False episode	52,2	50,3	49,8	50,9	47,6	49,3

Заключение

Результат работы имплементаций стандартных версий DQN и PPO алгоритмов демонстрируют способность решать поставленную задачу, тем не менее являются более вычислительно затратными. В сравнении с ними, в последующих имплементациях PPO существенно сокращается время обучения. В каждой из них исследуется применение ряда эвристик, таких как: изменение модели сети в PPOv1, параллелизация сред для ускорения сбора траекторий в PPOv2, модификации PPOv3, а также добавление механизма внимания в четвертой реализации PPO. Все реализованные агенты, как оригинальные DQN и PPO, так и вариации v1-v4 последнего, решают поставленную задачу на уровне ~50 % точности за 1000 пройденных тестовых эпизодов.

Текущая работа представляет собой продолжающееся исследование и предполагает проведение экспериментов с модификациями PPO, улучшениями общего процесса обучения и эффективности агентов.

Логичным продолжением работ также является замена среды для проведения симуляций более комплексных задач. Потенциально это может быть и разработка авторской системы симуляции среды, которая может открыть новые направления исследований в области обучения с подкреплением.

СПИСОК ЛИТЕРАТУРЫ/REFERENCES

1. Arulkumaran K., Deisenroth M.P., Brundage M., Bharath A.A. Deep reinforcement learning: a brief survey. *IEEE Signal Processing Magazine*, 2017, vol. 34, no. 6, pp. 26–38. DOI: 10.1109/MSP.2017.2743240
2. Hao-nan Wang, Ning Liu, Yi-yun Zhang, Da-wei Feng, Feng Huang, Dong-sheng Li, Yi-ming Zhang. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 2020, vol. 21, no. 12, pp. 1726–1744. DOI: https://doi.org/10.1631/FITEE.1900533
3. Wang X., Wang S., Liang X., Zhao D., Huang J., Xu X., Dai B., Miao Q. Deep reinforcement learning: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022, vol. 35, Iss. 4, pp. 5064–5078. DOI: 10.1109/TNNLS.2022.3207346
4. *Bullet Real-Time Physics Simulation*. Available at: https://pybullet.org/ (accessed: 23 June 2024).
5. *Gazebo*. Available at: https://gazebosim.org/ (accessed: 23 June 2024).
6. Zeng A., Song Sh., Lee J., Rodriguez A., Funkhouser Th. TossingBot: learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 2020, vol. 36, no. 4, pp. 1307–1319.
7. Zeng A., Florence P., Tompson J., Welker S., Chien J., Attarian M., Armstrong T., Krasin I., Duong D., Sindhwani V., Lee J. Transporter networks: rearranging the visual world for robotic manipulation. *4th Conference on Robot Learning (CoRL 2020)*. Cambridge MA, USA, 2020. pp. 726–747. Available at: https://transporternets.github.io/ (accessed: 23 June 2024).
8. Kovač L., Farkaš I. Safe reinforcement learning in a simulated robotic arm. *International Conference on Artificial Neural Networks*. Cham, Springer Nature Switzerland, 2023. pp. 585–589.
9. Aumjaud P., McAuliffe D., Rodríguez-Lera F.J., Cardiff Ph. Reinforcement learning experiments and benchmark for solving robotic reaching tasks. *Advances in Physical Agents II. WAF 2020. Advances in Intelligent Systems and Computing*. Cham, Springer International Publ., 2021, vol. 1285, pp. 318–331. DOI: https://doi.org/10.1007/978-3-030-62579-5_22

10. Kumar S., Sampson H., Behera A. *Benchmarking deep reinforcement learning algorithms for vision-based robotics*. DOI: https://doi.org/10.48550/arXiv.2201.04224 Available at: https://arxiv.org/abs/2201.04224 (accessed: 23 June 2024).
11. Chen H. Robotic manipulation with reinforcement learning, state representation learning, and imitation learning (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, no. 18, pp. 15769–15770.
12. Luo S., Kasaei H., Schomaker L. Accelerating reinforcement learning for reaching using continuous curriculum learning. *Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, Article 9207427. DOI: https://doi.org/10.1109/IJCNN48605.2020.9207427
13. Mnih V., Kavukcuoglu K., Silver D., Rusu A.A., Veness J., Bellemare M.G., Graves A., Riedmiller M., Fidjeland A.K., Ostrovski G., Petersen S., Beattie Ch., Sadik A., Antonoglou I., King H., Kumaran Dh., Wierstra D., Legg Sh., Hassabis D. Human-level control through deep reinforcement learning. *Nature*, 2015, vol. 518, pp. 529–533. DOI: https://doi.org/10.1038/nature14236
14. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. *Proximal policy optimization algorithms*. DOI: https://doi.org/10.48550/arXiv.1707.06347. Available at: https://arxiv.org/abs/1707.06347 (accessed: 23 June 2024).
15. kuka_diverse_object_gym_env.py. *bulletphysics*. Available at: https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/gym/pybullet_envs/bullet/kuka_diverse_object_gym_env.py (accessed: 23 June 2024).
16. OpenAi Spinning Up Algorithms. *OpenAI*. Available at: https://spinningup.openai.com/en/latest/user/algorithms.html (accessed: 23 June 2024).
17. Weng L. *A (Long) Peek into Reinforcement Learning*. Available at: https://lilianweng.github.io/posts/2018-02-19-rl-overview/#deep-q-network (accessed: 23 June 2024).
18. Actor-critic algorithm in reinforcement learning. *Geeksforgeeks*. Available at: https://www.geeksforgeeks.org/actor-critic-algorithm-in-reinforcement-learning/ (accessed: 23.06.2024)
19. Weng L. *Policy Gradient Algorithms*. Available at: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/#ppo (accessed: 23 June 2024).
20. Kingma D.P., Ba J. *Adam: a method for stochastic optimization*. DOI: https://doi.org/10.48550/arXiv.1412.6980. Available at: https://arxiv.org/abs/1412.6980 (accessed: 23 June 2024).
21. multiprocessing – process-based parallelism. *Python Software Foundation*. Available at: https://docs.python.org/3.10/library/multiprocessing.html (accessed: 23 June 2024).
22. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I. *Attention is all you need*. DOI: https://doi.org/10.48550/arXiv.1706.03762. Available at: https://arxiv.org/abs/1706.03762 (accessed: 23.06.2024)

Информация об авторах

Никита Евгеньевич Залогин, магистрант, инженер отделения информационных технологий Инженерной школы информационных технологий и робототехники, Национальный исследовательский Томский политехнический университет, Россия, 634050, г. Томск, пр. Ленина, 30; nez4@tpu.ru; https://orcid.org/0009-0003-1611-1637

Дмитрий Сергеевич Григорьев, старший преподаватель отделения информационных технологий Инженерной школы информационных технологий и робототехники, Национальный исследовательский Томский политехнический университет, Россия, 634050, г. Томск, пр. Ленина, 30; trygx@tpu.ru; https://orcid.org/0000-0001-8810-6691

Поступила в редакцию: 02.07.2024
Поступила после рецензирования: 03.09.2024
Принята к публикации: 30.09.2024

Information about the authors

Nikita E. Zalogin, Master Student, Engineer, National Research Tomsk Polytechnic University, 30, Lenin avenue, Tomsk, 634050, Russian Federation; nez4@tpu.ru; https://orcid.org/0009-0003-1611-1637

Dmitriy S. Grigorev, Senior Lecturer, National Research Tomsk Polytechnic University, 30, Lenin avenue, Tomsk, 634050, Russian Federation; trygx@tpu.ru; https://orcid.org/0000-0001-8810-6691

Received: 02.07.2024
Revised: 03.09.2024
Accepted: 30.09.2024