|  |  |  |
| --- | --- | --- |
| Федеральное государственное автономное образовательное учреждение  высшего образования «Национальный исследовательский  Нижегородский государственный университет им. Н.И. Лобачевского» НИИМ Нижегородского университета | | |
|  | | **УТВЕРЖДАЮ**  Профессор кафедры  ИАНИ ННГУ, д.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Старостин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |
| Научно-технический отчет  **РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**  **«Разработка агента в OpenAI Gym с использованием**  **методик обучения с подкреплением»** | | |
|  |  | |
|  |  | |
|  |  | |

Нижний Новгород

2021

СОДЕРЖАНИЕ

[Введение 3](#_Toc73190984)

[1. Содержательная постановка задачи 4](#_Toc73190985)

[1.1. Описание среды 4](#_Toc73190986)

[2. Входные данные 5](#_Toc73190987)

[3. Выходные данные 6](#_Toc73190988)

[3.1. Запуск обучения моделей 6](#_Toc73190989)

[3.2. Запуск обученной модели 6](#_Toc73190990)

[4. Разработка моделей обучения 8](#_Toc73190991)

[4.1. Собственные модель обучения 8](#_Toc73190992)

[4.2. Q-Learning модель обучения 9](#_Toc73190993)

[4.3. Нейросетевая модель обучения 10](#_Toc73190994)

[5. Процесс обучения агента 12](#_Toc73190995)

[5.1. Системные сообщения 12](#_Toc73190996)

[5.2. Возможные ошибки во время обучения агента 12](#_Toc73190997)

[6. Результаты обучения и работы обученного агента 14](#_Toc73190998)

[Заключение 16](#_Toc73190999)

[Ссылки на источники 17](#_Toc73191000)

# Введение

В рамках данной исследовательской работе разрабатывается агент в OpenAI Gym с использованием методик обучения с подкреплением.

В рамках проекта рассмотрено несколько подходов обучения агента для выполнения его задачи в среде LunarLander-v2, таких как: собственная модель обучения, Q-Learning модель обучения и нейросетевая модель обучения.

# Содержательная постановка задачи

Задачей данного проекта является изучение и реализация алгоритмов, которые будут осуществлять обучение посадке модуля в среде LunarLander-v2, при этом максимизируя количество очков награды.

## Описание среды

Среда представляет собой 2D приложение с летающим модулем и поверхностью с выделенной площадкой для посадки. Модуль необходимо посадить на площадку, избегая его крушения. В начале каждого эпизода модулю сообщается случайный импульс. На модуль можно воздействовать путем включения одного из двигателей за такт. Очки награды начисляются за успешное приземление модуля на поверхность. Вычитаются очки за каждое использование двигателей (по -0.03 за боковые и -0.3 за главный) и за жёсткое приземление (приземление с большой скоростью и/или крушение).

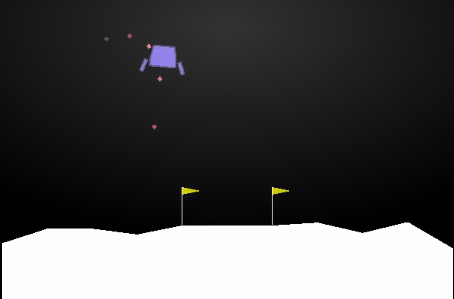


Рис 1. Внешний вид среды LunarLander-v2

Целью данной игры является успешная посадка спускаемого модуля на поверхность. Посадка считается успешной, если: спускаемый модуль приземлился мягко на ножки на посадочную площадку (между флажками), а также затратил малое количества топлива. Хорошее значение награды в таком случае будет больше 200.

# Входные данные

Входными данными является сама среда LunarLander-v2 и механизмы взаимодействия с ней через код путём пользовательского ввода. Среда реализована на языке Python, и её полный код реализации находится в открытом доступе [[1](#_Ссылки_на_источники)]. Описание основного функционала среды на русском языке находится в отчёте-аннотации.

# Выходные данные

Выходными данными работы являются графики обучения агента разными моделями, гистограммы результатов работы обученной модели, видеозаписи фрагментов обучения агента, а также веса нейросети, полученные в ходе обучения. Скачать репозиторий проекта можно по ссылке [[2](#_Ссылки_на_источники)].

## Запуск обучения моделей

Для того чтобы запустить обучение агента на ОС Windows, необходимо:

* открыть командную строку cmd и перейти в папку {Ваш путь}/LunarLanderProject/Code командой cd или же зайти в эту папку через проводник и открытый через PowerShell нажатием Shift+правая кнопка мыши;
* далее ввести команду «python learn\_models.py», выбрать необходимую модель и указать число запусков.

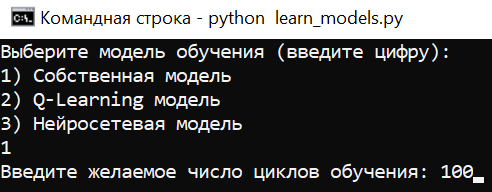


Рис 2. Пример запуска обучения агента через cmd

Также можно запустить данный файл, открыв его через среду разработки, например, Spyder.

## Запуск обученной модели

Для того чтобы запустить демонстрацию работы обученного агента на ОС Windows, необходимо:

* открыть командную строку cmd и перейти в папку {Ваш путь}/LunarLanderProject/Code командой cd или же зайти в эту папку через проводник и открытый через PowerShell нажатием Shift+правая кнопка мыши;
* далее ввести команду «python lunar\_lander.py» и указать число запусков.
* указать, сохранять ли диаграммы по результатам запусков.

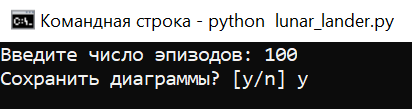


Рис 3. Пример запуска демонстрации обученного агента через cmd

Также можно запустить данный файл, открыв его через среду разработки, например, Spyder.

В проекте собран запускаемый файл lunar\_lander.exe для того, чтобы любой пользователь мог посмотреть результат обучения модели без установки Python на своё устройство. В папке «Trained Lander» необходимо найти файл lunar\_lander.exe и создать его ярлык, как в примере ниже. Скачать архив с приложением можно по ссылке [[3](#_Ссылки_на_источники)].

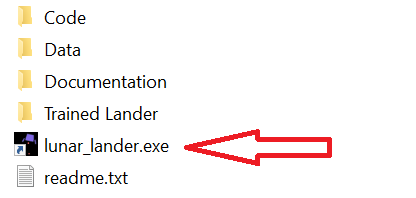


Рис 4. Пример создания ярлыка приложения

При запуске пользователю нужно лишь указать желаемое количество эпизодов игры и возможность сохранения графиков результата. При этом результаты запусков будут выводиться в консоль.



Рис 5. Результат запуска приложения

# Разработка моделей обучения

## Собственные модель обучения

В качестве самобытных подходов к обучению были рассмотрены табличная модель и модель, основанная на RDF (Random Decision Forest).

Табличная модель представляет собой модель обучения, основанную на полном переборе таблицы состояний приземляемого модуля, действий над ним и оценки его действий. Выбираются лучшие действия, в соответствие с наградой за траекторию, из строк таблицы на основании евклидового расстояния между состояниями.

Во второй модели схожая логика, только поиск действий происходит с помощью RDF, который построен на основе той же таблицы.

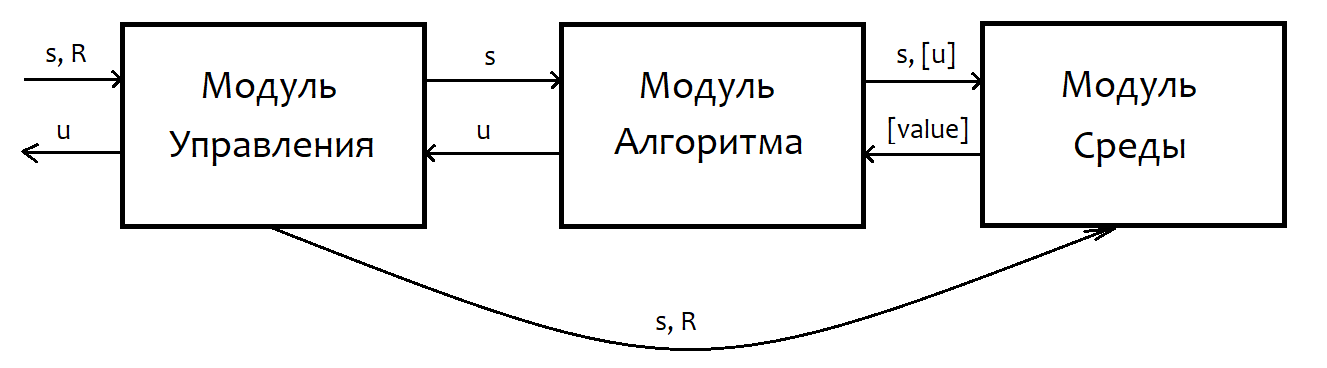


Рис 6. Структура архитектуры собственной модели

**Блок «Модуль управления»** принимает на вход текущее состояние (s) и сохраняет его в текущую траекторию. С последним состоянием траектории блок принимает итоговую «ценность» этой траектории (R) и отправляет эту траекторию в «Модуль среды». В задачи блока также входит отправить в «Модуль алгоритма» текущее состояние и получить от него управление (u), которое он вернёт назад клиенту вызова.

**Блок «Модуль алгоритма»** принимает на вход текущее состояние (s) и запрашивает у блока «Модуль среды» предсказания по текущему состоянию с каждым из возможных действий (u). Из полученных ответов выбирается наиболее выгодный в плане действия, и это действие возвращается блоку «Модуль управления».

**Блок «Модуль среды»** содержит в себе методы предсказания на базе RDF (Random Decision Forest) и на базе таблицы. RDF изначально обучается на таблице случайных запусков. В ходе работы алгоритма блок «Модуль управления» посылает набор траекторий (s, R), по мере накопления определённого количества которых данные добавляются в таблицу, и RDF переобучается. Метод предсказания на базе RDF по набору состояний (s) и действий (u) возвращает блоку «Модуль алгоритма» набор предсказанных значений награды (value). Метод предсказания по таблице по набору состояний (s) ищет в таблице ближайшие по Евклидову расстоянию состояния и выдаёт действие (u) при максимальном значении награды.

## Q-Learning модель обучения

Данная модель обучения представляет собой один из классических подходов к обучению с подкреплением (Reinforcement Learning), применяемый при агентном подходе.

На основе получаемого от среды вознаграждения и функцию расчёта полезности Q формируется таблица, из которой по текущему состоянию модуля можно выбрать наиболее выгодное действие с точки зрения функции полезности. По мере обучения добавляются новые состояния и корректируются значения в таблице, что впоследствии дает возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия модуля со средой.

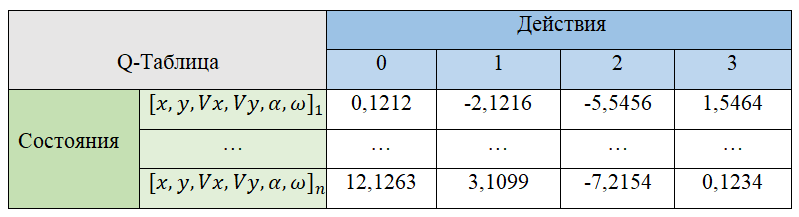


Рис 7. Пример заполнения Q-Таблицы

Классическая схема алгоритма Q-Learning приведена ниже [[4](#_Ссылки_на_источники)].

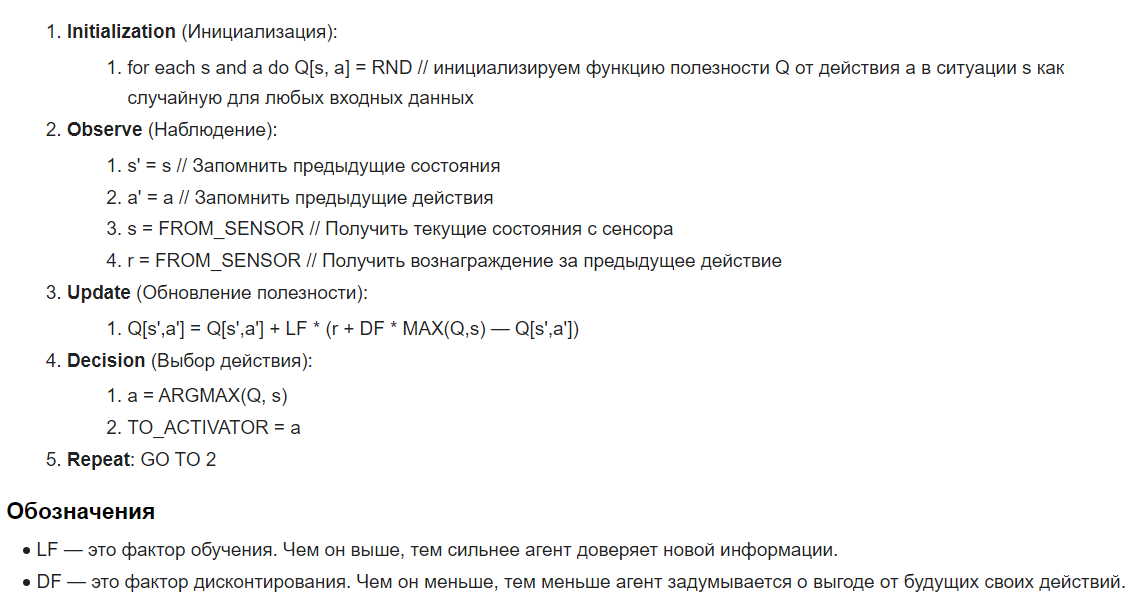


Рис 8. Схема Q-Learning

## Нейросетевая модель обучения

Данная модель обучения состоит из двух подсетей: «Актера» и «Критика». Модель «Актер» выполняет задачу изучения того, какие действия следует совершить в конкретном наблюдаемом состоянии окружающей среды. В рамках среды LunarLander-v2 «Актеру» поступает на вход 8 значений (состояние среды), а выход выдаёт вектор из 4 значений (вероятности выбора каждого действия). После завершения этих действий и получения очков награды, качество решений «Актера» оцениваются «Критиком». Таким образом, данные подсети дополняют друг друга, основываясь на вердиктах.

В качестве функции потерь используется PPO (Proximal Policy Optimization). Основная идея заключается в том, что после обновления новая политика не должна сильно отличаться от старой. Для этого PPO использует отсечение, чтобы избежать слишком больших обновлений. Это приводит к меньшему разбросу в обучении за счет некоторой предвзятости, но обеспечивает «более плавное» обучение, а также гарантирует, что агент не начнёт совершать бессмысленные действия.

Идея такого подхода взята из статьи об обучении агента в среде LunarLander-v2 [[5](#_Ссылки_на_источники)].

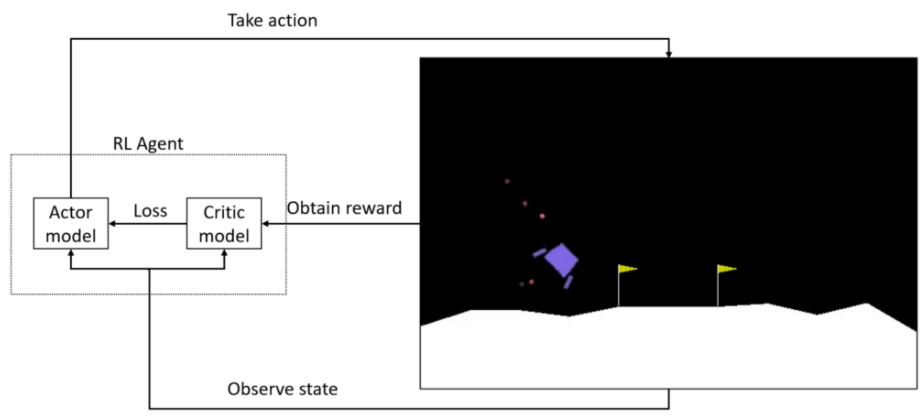


Рис 9. Структура модели «Актёр – Критик»

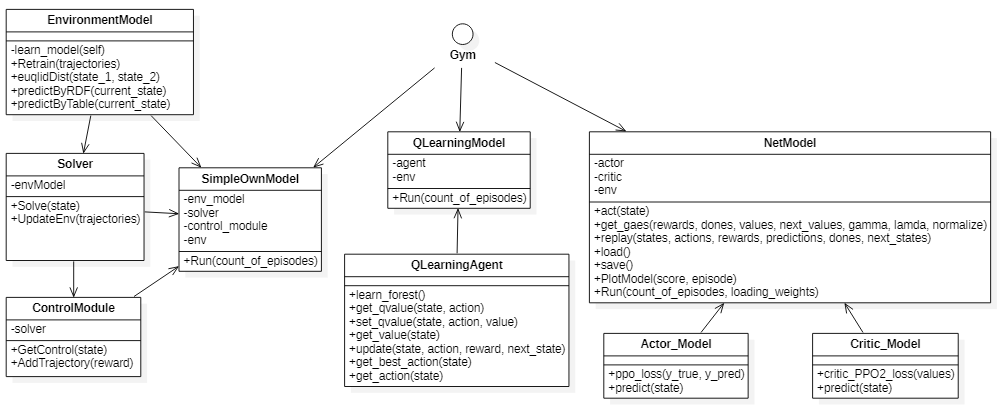


Рис 10. Общая схема архитектуры системы

# Процесс обучения агента

## Системные сообщения

В случае успешного запуска агента, откроется окно с рендером среды LunarLander.

При запуске обучения агента из среды разработки справа в окне консоли будет вестись вывод результатов эпизодов: текущий номер эпизода, награда и среднее. При запуске через cmd или PowerShell аналогичный вывод будет осуществлять прямо в этом же консоли. При запуске обучения собственной моделью дополнительно будут выводиться сообщения о добавлении траекторий и переобучении дерева.

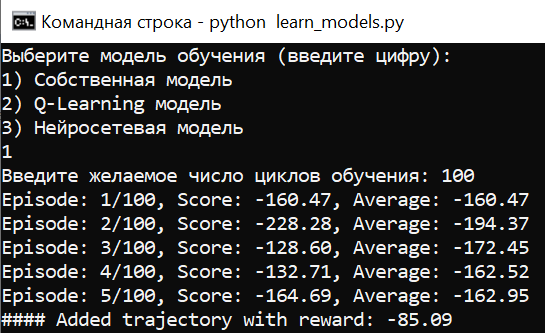


Рис 11. Пример вывода в консоль при запуске обучения собственной моделью

При запуске демонстрации работы обученного агента вывод осуществляется по той же аналогии, только немного видоизменён.

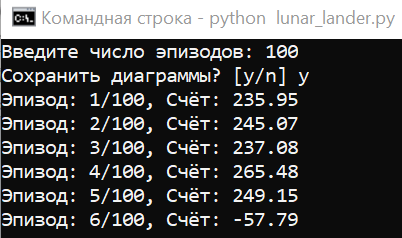


Рис 12. Пример вывода в консоль при запуске обученного агента

## Возможные ошибки во время обучения агента

В процессе эксплуатации агента аварийные ситуации, выражающиеся в отсутствии отклика, могут возникать по следующим причинам:

* Потеря питания ПК;
* Принудительное закрытие приложения во время обучения агента;
* Работа на ПК, характеристики которого не удовлетворяют системным требованиям.

При потере питания, необходимо заново запустить ПК и перезапустить среду разработки. В этом случае данные обучения могут быть утеряны.

В случае работе на ПК, характеристики которого не удовлетворяют системным требованиям, возможно подвисание окна, в котором происходит визуализация среды, а также возможен «вылет» из программы, с дальнейшей потерей данных.

При желании пользователя досрочно завершить обучение – необходимо перезапустить ядро Python. Для этого необходимо на верхней панели найти вкладку «Consoles», открыть ее и нажать «Interrupt kernel».

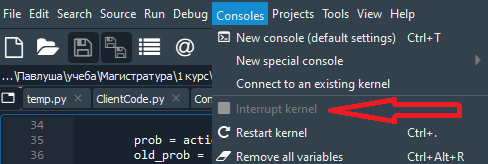


Рис 13. Иллюстрация перезагрузки ядра в среде разработки Spyder

Далее необходимо нажать «Restart kernel» (находится под Interrupt kernel). Ядро перезапустится, однако данные обучения будут утеряны.

Чтобы досрочно завершить работу при запуске через консоль, необходимо нажать Ctrl+C.

# Результаты обучения и работы обученного агента

Все модели обучения были запущены на 1300 эпизодов. Результаты обучения представлены ниже.

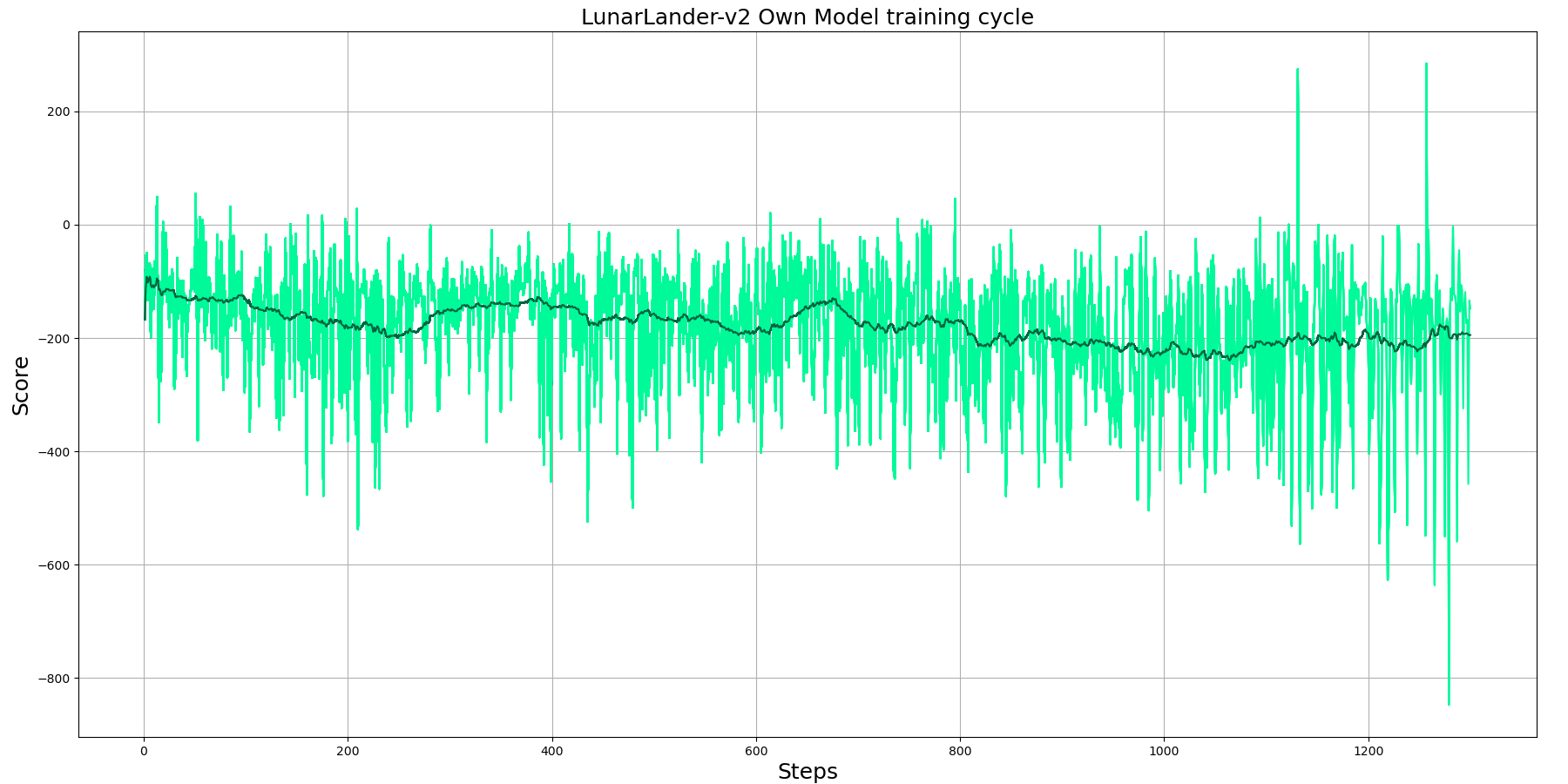


Рис 14. График результатов обучения агента собственной моделью обучения

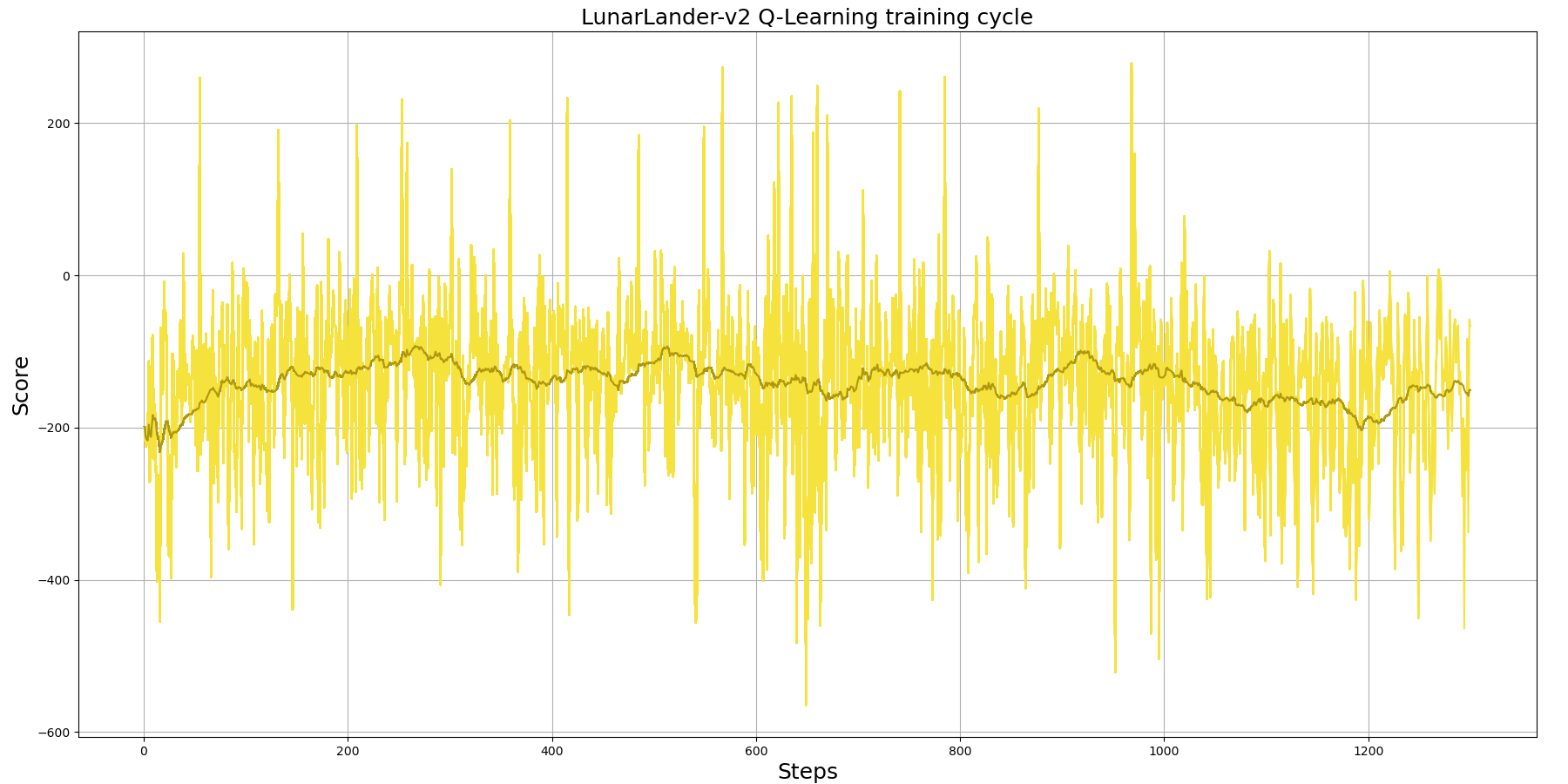


Рис 15. График результатов обучения агента Q-Leaning моделью обучения

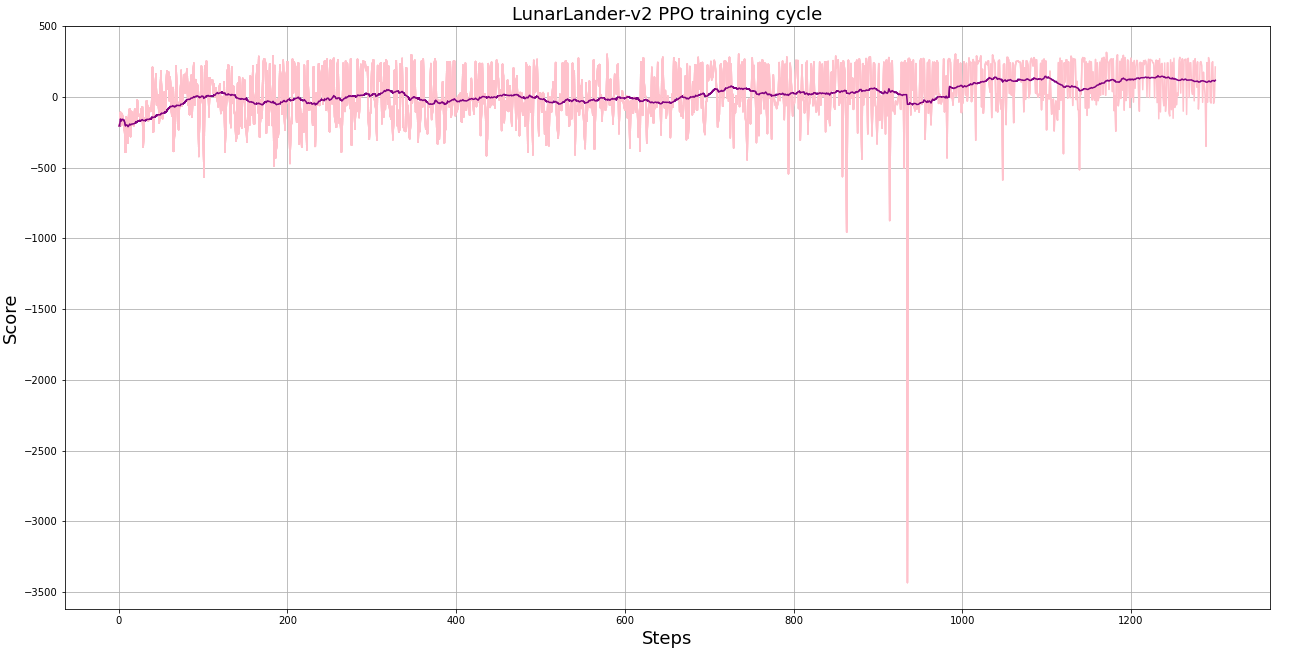


Рис 16. График результатов обучения агента нейросетевой моделью обучения

Ниже представлены результаты работы агента, обученного нейросетью, на 100 запусках.

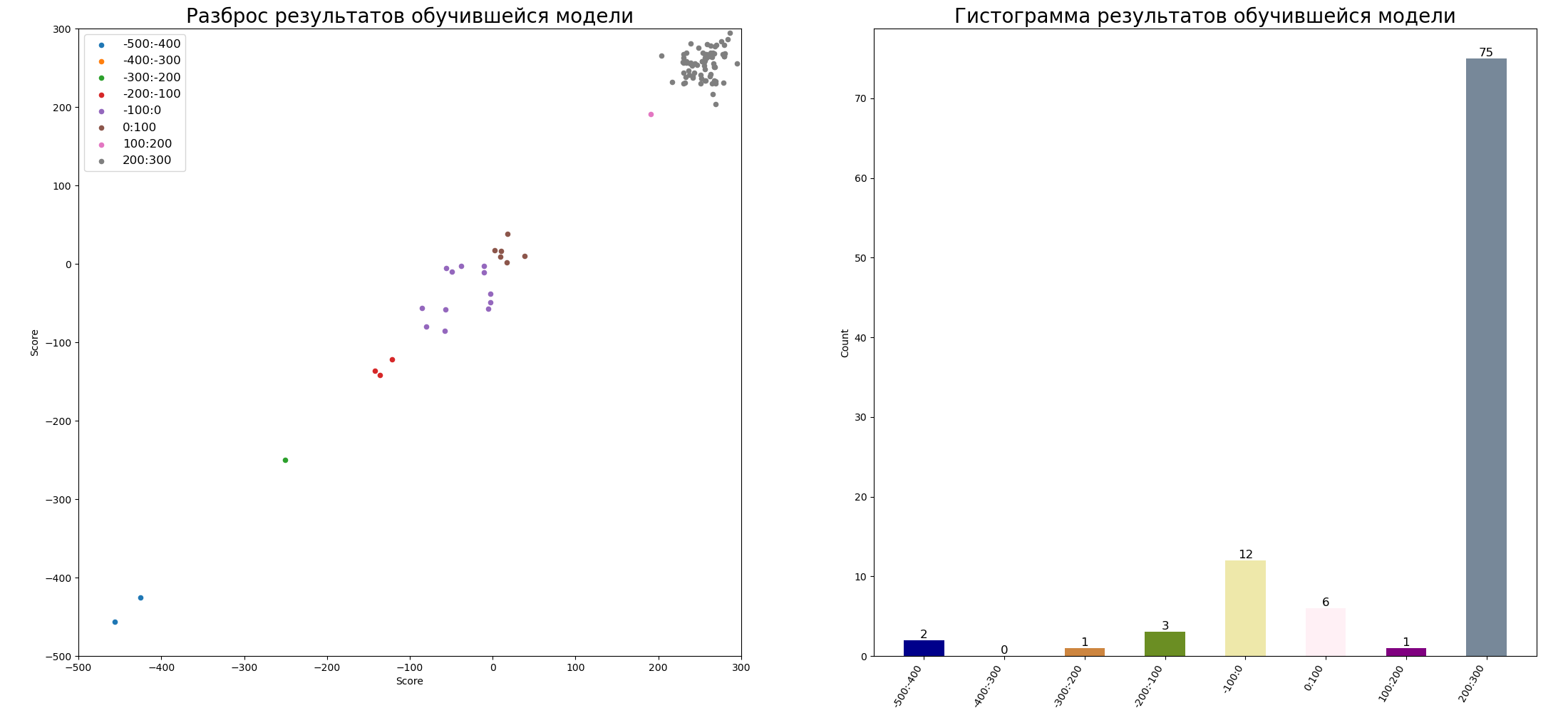


Рис 17. Гистограммы результатов работы обученного агента

# Заключение

Была поставлена задача обучения агента в среде LunarLander-v2 при помощи нескольких моделей обучения с целью выполнения его задачи. Агент получил возможность обучаться при помощи одной из выбранных моделей.

По результатам обучения были установлены следующие факты:

* Собственные модели обучения оказались недостаточно эффективны по ряду признаков: по памяти в работе RDF, по времени поиска по таблице, а также по невозможности эффективной оценки действий такими методами. На графике результатов видно, что средняя награда в районе -200, что очень мало.
* Q-Learning модель обучения оказалась также малоэффективной. По поведению агента и результатам обучения можно сделать вывод, что он обучается, но средний показатель награды в процессе обучения не возрастает и крутится в районе -150. Скорее всего, это связано с тем, что классический Q-Learning алгоритм не подходит для нашего агента, так как у него много параметров.
* Нейросетевой подход к обучению оказался эффективным, и в ходе обучения была заметна тенденция увеличения среднего значения награды, которая после 1300 эпох достигла среднего значения 150. Хороший результат обучения достигнут в силу выбранного подхода к обучению: используется эффективная функция потерь PPO и 2 взаимодополняющие друг друга нейросети. Однако данный подход потребовал очень долгого времени обучения в силу переобучения двух нейросетей.

По итогу обучения получены веса нейросети, которые были использованы для демонстрации работы агента через созданное приложение. На гистограмме результатов обученной модели можно видеть, что эффективность агента составляет 70-75%.

# Ссылки на источники

1. Код среды LunarLander-v2 на GitHub: <https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py>
2. Репозиторий проекта: <https://github.com/timka-rabbit/LunarLanderProject>
3. Архив с приложением: <https://disk.yandex.ru/d/PbVVQeIE-j8RiA>
4. Q-обучение [Wikipedia]: <https://ru.wikipedia.org/wiki/Q-обучение>
5. Обучение агента на основе нейросети «Актер - Критик»: <https://pylessons.com/LunarLander-v2-PPO/>