МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по проекту

по дисциплине «Обучение с подкреплением»

Тема: DQN #1

Студент гр. 0306 _____ Гудов Н.Р.

Студент гр. 0306 _____ Подопригора И.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург

2025

Цель работы.

Ознакомиться с особенностями различных версий DQN на разных окружениях для тестирования.

Задание.

Необходимо реализовать и сравнить между собой следующие версии DQN:

- Double Q-learning
- Prioritized replay
- Dueling networks

В качестве окружения для тестирования:

- LunarLander-v3
- Mountain-Car

Выполнение работы.

Реализация DQN.

Базовый DQN — это алгоритм, сочетающий Q-learning с глубокими нейронными сетями для обучения агентов в сложных средах с большими пространствами состояний. По сравнению с классическим Q-learning, DQN решает проблему масштабируемости за счёт использования нейросетей для аппроксимации Q-функции. На вход сеть получает текущее состояние среды, на выходе выдаёт предсказанные Q-значения для всех возможных действий. Для стабилизации обучения DQN использует буфер воспроизведения опыта, который обеспечивает обучение на случайных мини-батчах из этого буфера. Что разрушает временные корреляции между последовательными состояниями и позволяет многократно использовать ценный опыт. Для баланса между исследованием и эксплуатацией с вероятностью ε выбирается случайное действие, иначе – действие с максимальным Q-значением.

Double Q-learning

DDQN решает проблему переоценки Q-значений, которая возникает из-за использования одной и той же сети для выбора и оценки действий. В стандартном DQN максимизация Q-значений приводит к систематическому завышению оценок, особенно в определенных средах, что может замедлять обучение и ухудшать итоговую политику. DDQN устраняет положительное смещение, что делает обучение более стабильным и улучшает производительность агента.

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q\left(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t'\right)$$

Prioritized replay

В классическом DQN опыт из буфера воспроизведения выбирается равномерно, что можеть быть неэффективно, поскольку некоторые примеры содержат более ценную информацию для обучения. Prioritized Replay ускоряет обучение, уделяя больше внимания переходам, где текущие предсказания Q-значений наиболее неточны. В результате агент быстрее учится на критически важных моментах, что может повысить эффективность использования данных и сократить время обучения.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \qquad w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta$$

Dueling networks

Стандартный DQN оценивает Q-значения напрямую, не разделяя вклад состояния и действий, что может быть неэффективно в средах, где ценность состояния мало зависит от конкретного действия. Dueling Networks решают эту проблему, разделяя Q-функцию на два компонента: ценность состояния (V) и преимущество действий (A). Таким образом повышается устойчивость оценок в средах с избыточными или малозначимыми действиями.

$$Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\beta) + \left(A(s,a,\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A\left(s,a';\theta,\alpha\right)\right)$$

LunarLander-v3.

Состояние представляет собой 8-мерный вектор: координаты посадочного модуля в x& y, его линейные скорости в x& y, его угол, его угловая скорость и два логических значения, которые представляют, находится ли каждая нога в контакте с землей или нет.

После каждого шага дается награда. Общая награда эпизода — это сумма наград за все шаги в этом эпизоде.

За каждый шаг награда:

- увеличивается/уменьшается по мере приближения/удаления посадочного модуля к посадочной площадке.
- увеличивается/уменьшается по мере того, как медленнее/быстрее движется посадочный модуль.
- уменьшается по мере наклона посадочного модуля (угол не горизонтальный).
- увеличивается на 10 очков за каждую ногу, которая находится в контакте с землей.
- уменьшается на 0,03 очка в каждом кадре, когда работает боковой двигатель.
- уменьшается на 0,3 балла за каждый кадр работы основного двигателя.

В эпизоде дополнительная награда в размере -100 или +100 очков за падение или благополучную посадку соответственно. Эпизод считается решением, если он набрал не менее 200 баллов.

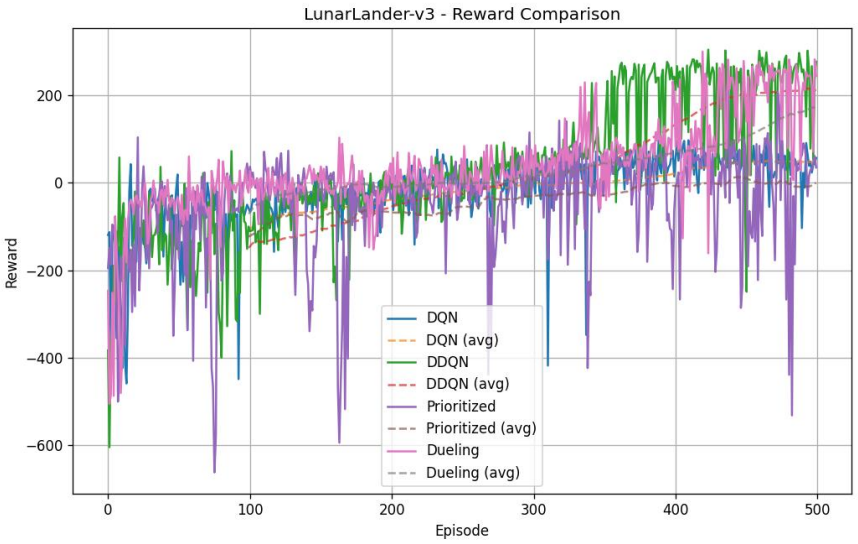

Рисунок 1. Награды в LunarLander-v3.

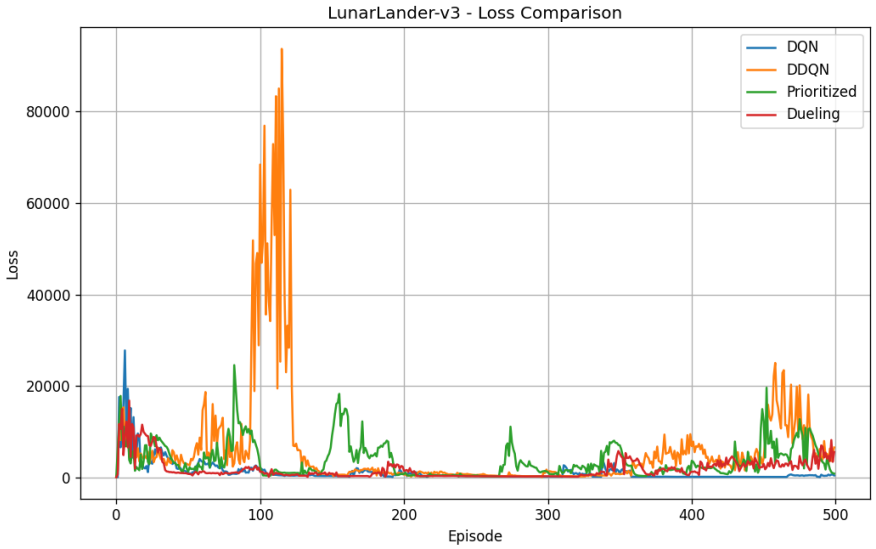

Рисунок 2. Потери в LunarLander-v3.

По графикам можно заметить что только Dueling и DDQN достигли успешного решения. К 500 эпизодам все реализации держались около значения 100, что означало бы успешную посадку.

Тем не менее хуже всего показал себя Prioritized replay, чаще других получая худшую награду. Такое поведение может быть связано с тем, что агент фокусируется на исправлении ошибок, а не на поиске оптимальной стратегии, а так же на зацикливании на одних и тех же сценариях. В следствии это может вызвать, например, замедление изучения ключевых моментов, например, мягкой посадки.

MountainCar-v0.

Состояние представляет собой 2-мерный вектор: положение автомобиля вдоль оси x и скорость автомобиля.

Цель состоит в том, чтобы как можно быстрее добраться до флага, установленного на вершине правого холма, в связи с чем агент получает штраф в размере -1 за каждый временной шаг.

Начальное состояние: Положению автомобиля присваивается равномерное случайное значение в диапазоне [-0,6 , -0,4] . Начальная скорость автомобиля всегда присваивается 0.

Конец эпизода: Эпизод заканчивается, если происходит одно из следующих событий: Положение автомобиля больше или равно 0,5 (целевое положение на вершине правого холма) или длина эпизода — 200.

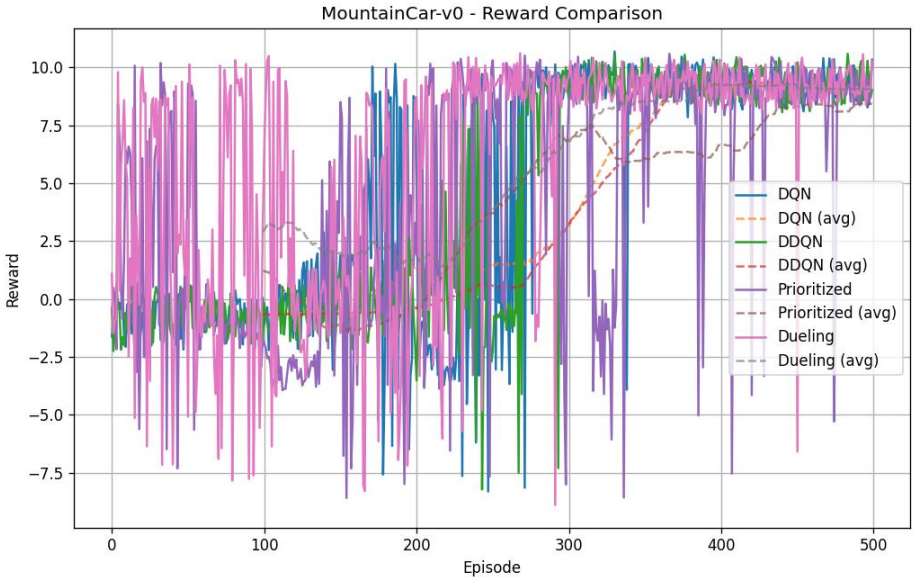

Рисунок 3. Награды в MountainCar-v0.

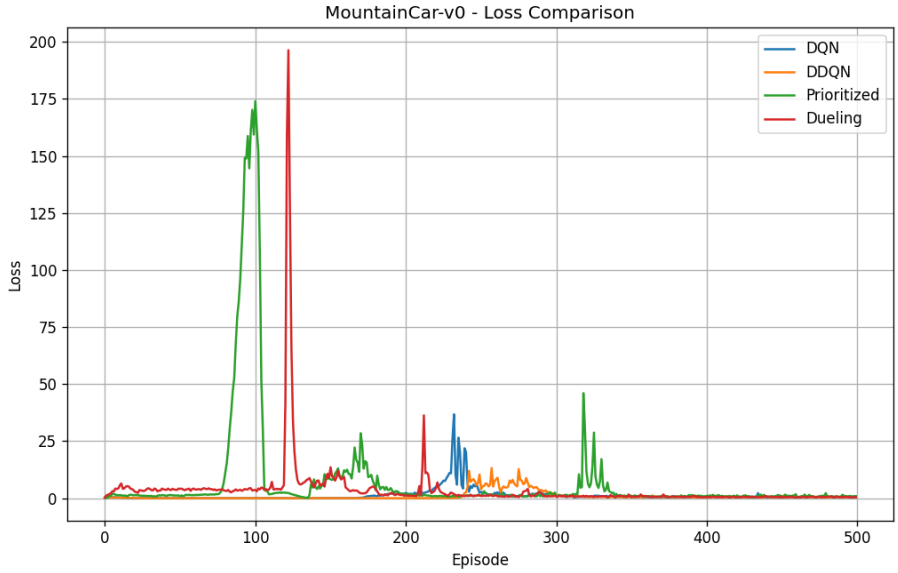

Рисунок 4. Потери в MountainCar-v0.

В среде MountainCar-v0 лучших результатов за меньшее время добиваются Dueling и Prioritized replay. Dueling в то же время добивается и наименьших наград, постоянно колеблясь от больших значений к малым. Prioritized replay через некоторое время понижает результаты, но не сильно уходит в отрицательные значения. Такое может быть вызвано тем, что агент не успевает адаптироваться к изменениям Q-значений или застреванием в локальных решениях. В среднем все реализации показывают эффективность после трехсот эпизодов.

Вывод.

В ходе работы рассмотрены модификации алгоритмов глубокого обучения с подкреплением DQN и осуществлена их практическая реализация с использованием фреймворка PyTorch. Проведены эксперимент по оценке эффективности указанных методов в нескольких средах. Полученные данные согласованы с теоретическими положениями. Результаты подтверждают, что современные модификации DQN в среднем превосходят базовый алгоритм, но их эффективность зависит от правильного выбора и настройки под конкретную задачу.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Исходный код размещен по ссылке: https://github.com/ngudow/RL_DQN-1