Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине
«Методы машинного обучения»
на тему

«Обучение на основе глубоких Q-сетей»

Выполнил:
студент группы ИУ5И-23М
Ван Тяньшо

Москва — 2024 г.

1. Цель лабораторной работы

Цель лабораторной работы: ознакомление с базовыми методами обучения с подкреплением на основе глубоких Q-сетей.

2. Задание

На основе рассмотренных на лекции примеров реализуйте алгоритм DQN.

В качестве среды можно использовать классические среды (в этом случае используется полносвязная архитектура нейронной сети).

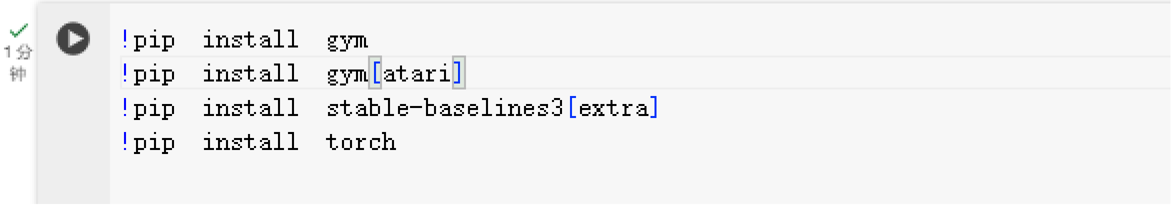
В качестве среды можно использовать игры Atari (в этом случае используется сверточная архитектура нейронной сети).

В случае реализации среды на основе сверточной архитектуры нейронной сети +1 балл за экзамен.

Выбранный мной набор данных: Используйте классическую среду управления (например, CartPole) и игровую среду Atari (например, Breakout).

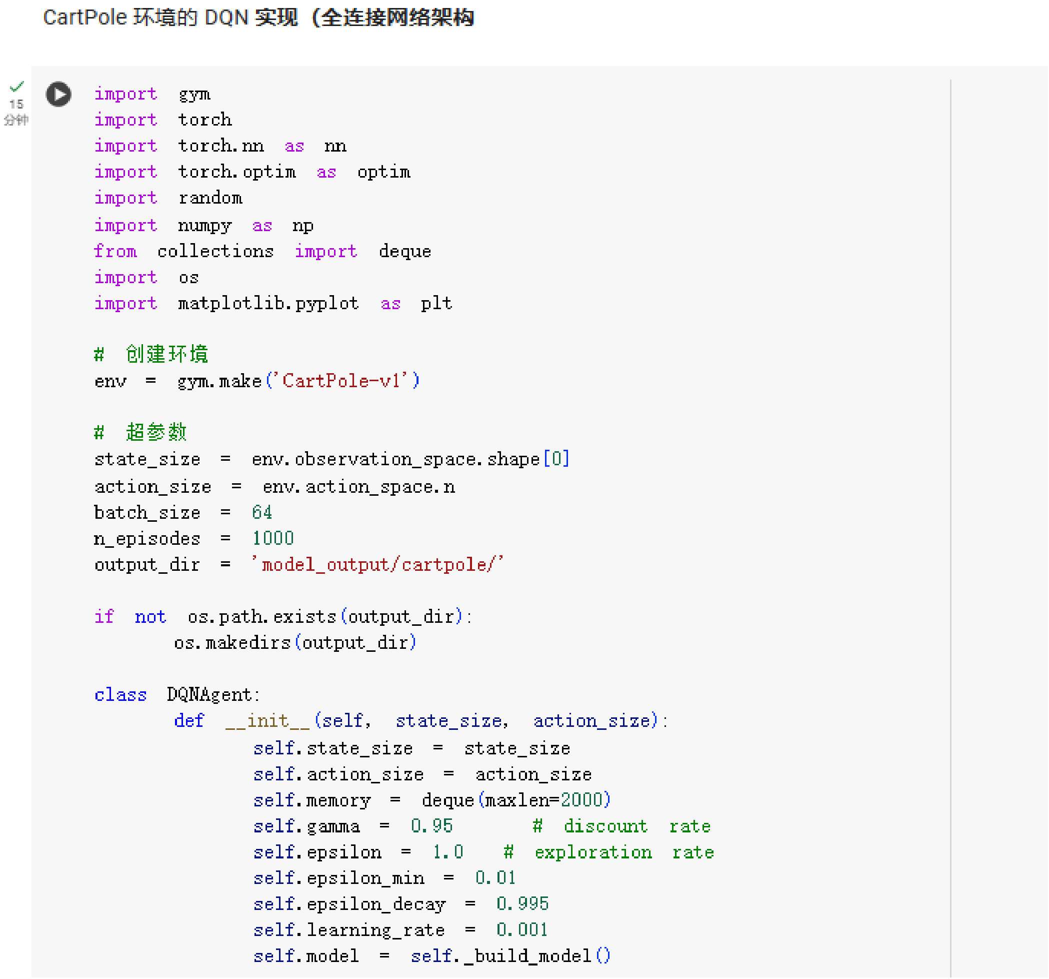
3. Текст программы

Шаг 1: Установите и импортируйте необходимые библиотеки



```
!pip install gym
!pip install gym[atari]
!pip install stable-baselines3[extra]
!pip install torch
```

Шаг 2: Реализация DQN среды CartPole (полностью подключенная сетевая архитектура)



CartPole 环境的 DQN 实现（全连接网络架构

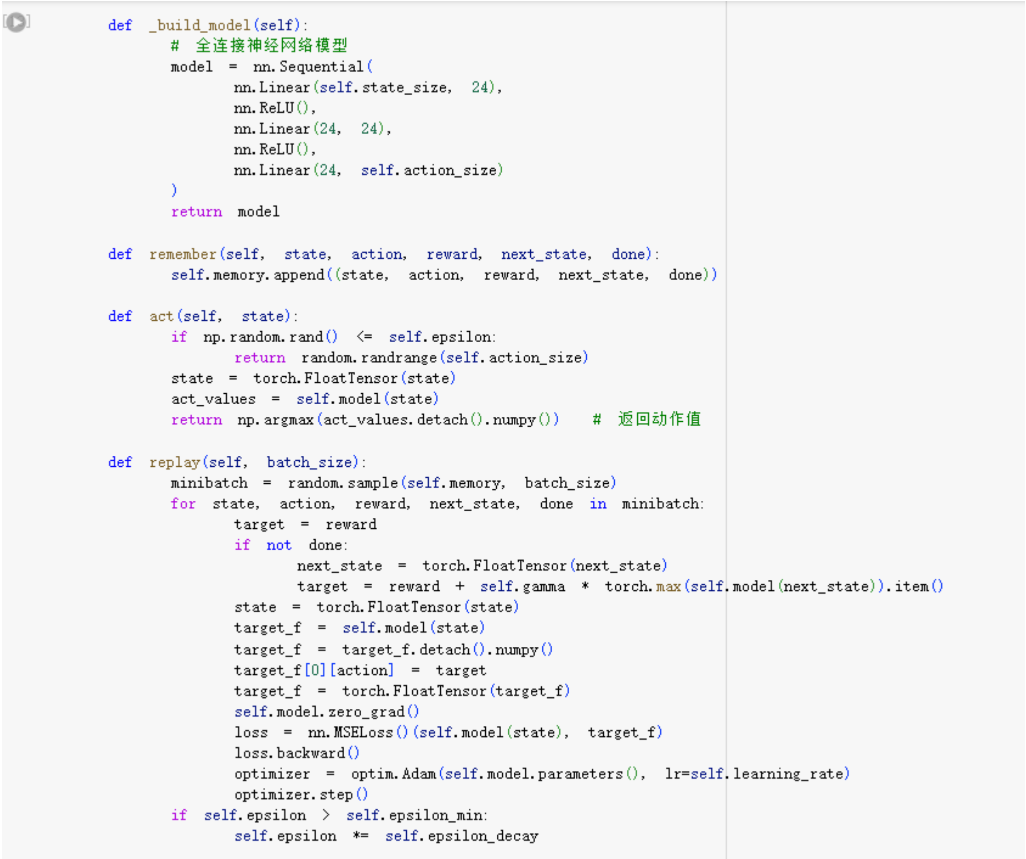
```
import gym
import torch
import torch.nn as nn
import torch.optim as optim
import random
import numpy as np
from collections import deque
import os
import matplotlib.pyplot as plt

# 创建环境
env = gym.make('CartPole-v1')

# 超参数
state_size = env.observation_space.shape[0]
action_size = env.action_space.n
batch_size = 64
n_episodes = 1000
output_dir = 'model_output/cartpole/'

if not os.path.exists(output_dir):
    os.makedirs(output_dir)

class DQNAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.memory = deque(maxlen=2000)
        self.gamma = 0.95    # discount rate
        self.epsilon = 1.0   # exploration rate
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995
        self.learning_rate = 0.001
        self.model = self._build_model()
```

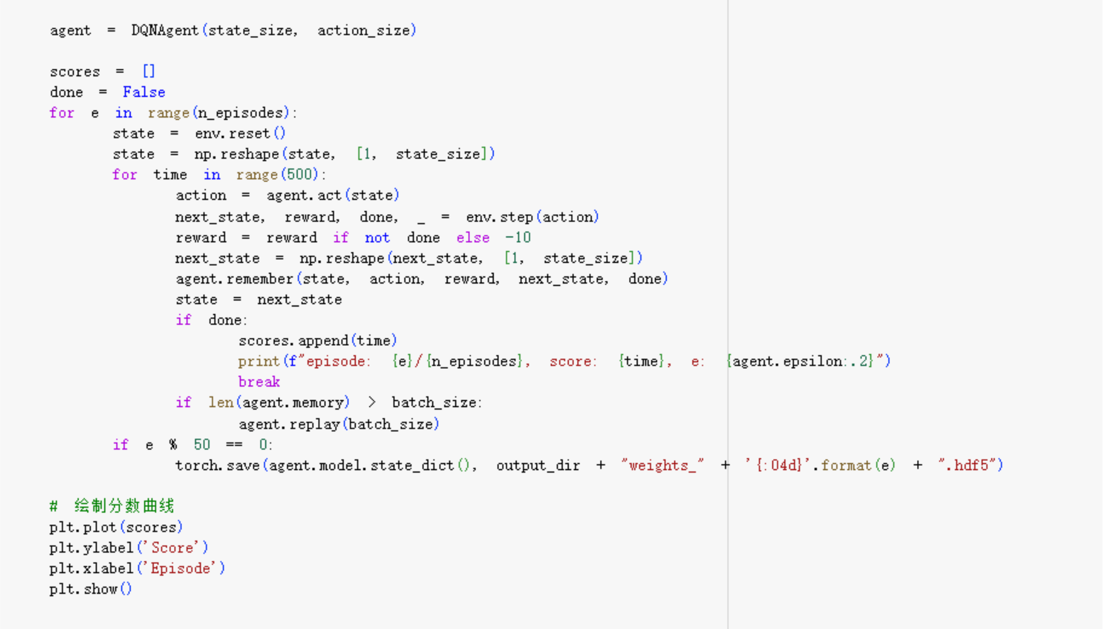


```python
    def _build_model(self):
        # 全连接神经网络模型
        model = nn.Sequential(
            nn.Linear(self.state_size, 24),
            nn.ReLU(),
            nn.Linear(24, 24),
            nn.ReLU(),
            nn.Linear(24, self.action_size)
        )
        return model

    def remember(self, state, action, reward, next_state, done):
        self.memory.append((state, action, reward, next_state, done))

    def act(self, state):
        if np.random.rand() <= self.epsilon:
            return random.randrange(self.action_size)
        state = torch.FloatTensor(state)
        act_values = self.model(state)
        return np.argmax(act_values.detach().numpy())  # 返回动作值

    def replay(self, batch_size):
        minibatch = random.sample(self.memory, batch_size)
        for state, action, reward, next_state, done in minibatch:
            target = reward
            if not done:
                next_state = torch.FloatTensor(next_state)
                target = reward + self.gamma * torch.max(self.model(next_state)).item()
            state = torch.FloatTensor(state)
            target_f = self.model(state)
            target_f = target_f.detach().numpy()
            target_f[0][action] = target
            target_f = torch.FloatTensor(target_f)
            self.model.zero_grad()
            loss = nn.MSELoss()(self.model(state), target_f)
            loss.backward()
            optimizer = optim.Adam(self.model.parameters(), lr=self.learning_rate)
            optimizer.step()
        if self.epsilon > self.epsilon_min:
            self.epsilon *= self.epsilon_decay
```

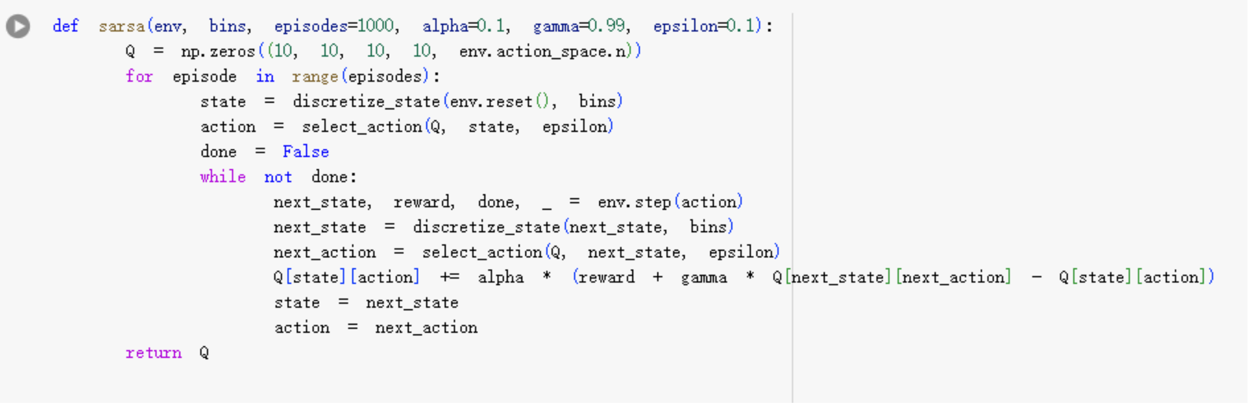


```python
agent = DQNAgent(state_size, action_size)

scores = []
done = False
for e in range(n_episodes):
    state = env.reset()
    state = np.reshape(state, [1, state_size])
    for time in range(500):
        action = agent.act(state)
        next_state, reward, done, _ = env.step(action)
        reward = reward if not done else -10
        next_state = np.reshape(next_state, [1, state_size])
        agent.remember(state, action, reward, next_state, done)
        state = next_state
        if done:
            scores.append(time)
            print(f"episode: {e}/{n_episodes}, score: {time}, e: {agent.epsilon:.2}")
            break
        if len(agent.memory) > batch_size:
            agent.replay(batch_size)
    if e % 50 == 0:
        torch.save(agent.model.state_dict(), output_dir + "weights_" + '{:04d}'.format(e) + ".hdf5")

# 绘制分数曲线
plt.plot(scores)
plt.ylabel('Score')
plt.xlabel('Episode')
plt.show()
```

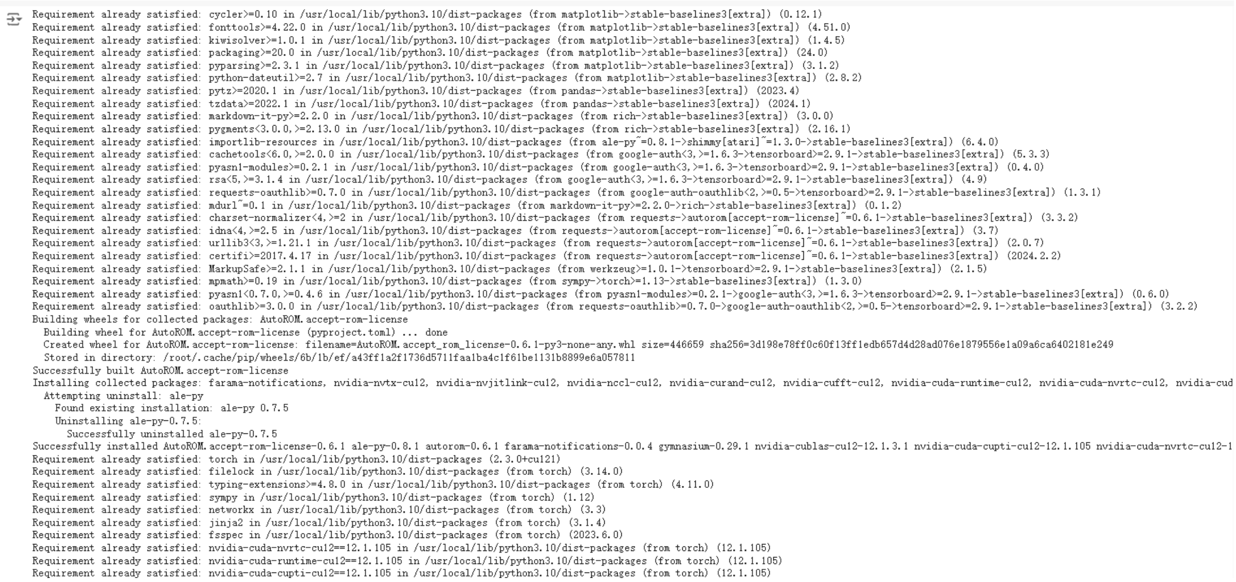
Шаг 3: Внедрение прорывной среды DQN (архитектура сверточной нейронной сети с визуализацией и сохранением видео)



```python
def sarsa(env, bins, episodes=1000, alpha=0.1, gamma=0.99, epsilon=0.1):
    Q = np.zeros((10, 10, 10, 10, env.action_space.n))
    for episode in range(episodes):
        state = discretize_state(env.reset(), bins)
        action = select_action(Q, state, epsilon)
        done = False
        while not done:
            next_state, reward, done, _ = env.step(action)
            next_state = discretize_state(next_state, bins)
            next_action = select_action(Q, next_state, epsilon)
            Q[state][action] += alpha * (reward + gamma * Q[next_state][next_action] - Q[state][action])
            state = next_state
            action = next_action
    return Q
```

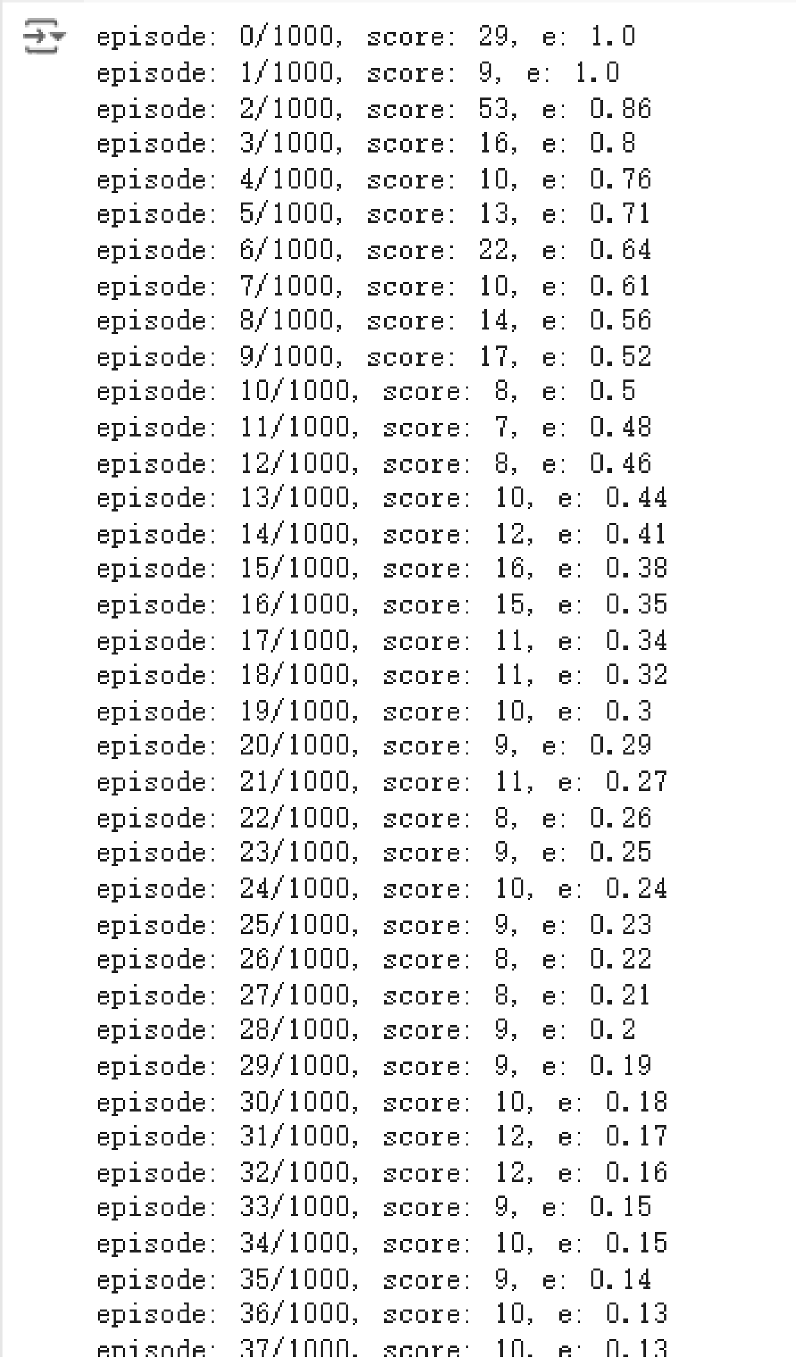
4. Экранные формы с примерами выполнения программы

Шаг 1: Установите и импортируйте необходимые библиотеки

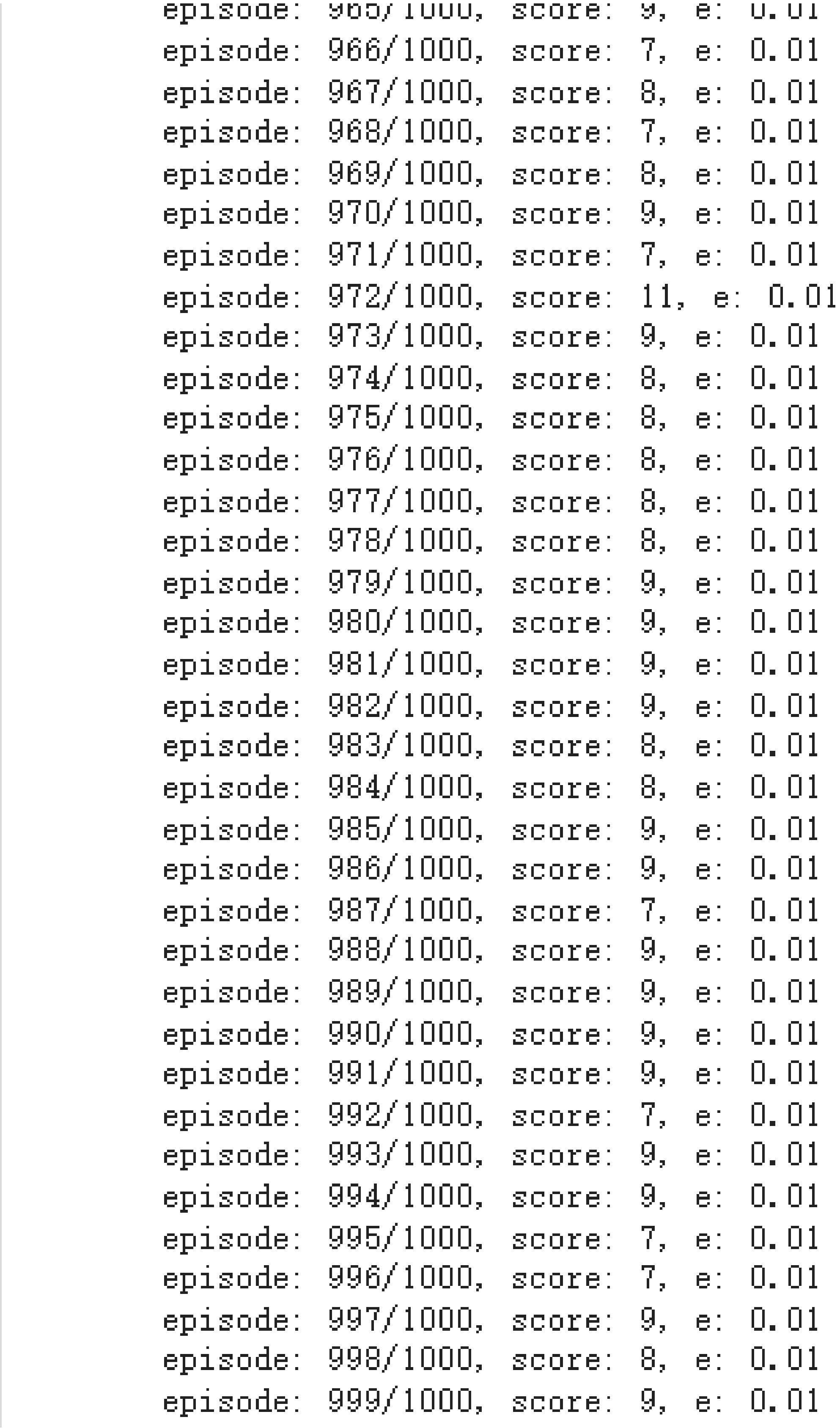


```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (24.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->stable-baselines3[extra]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->stable-baselines3[extra]) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->stable-baselines3[extra]) (2024.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->stable-baselines3[extra]) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->stable-baselines3[extra]) (2.16.1)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/dist-packages (from ale-py~=0.8.1->shimmy[atari]~=1.3.0->stable-baselines3[extra]) (6.4.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard>=2.9.1->stable-baselines3[extra]) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard>=2.9.1->stable-baselines3[extra]) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard>=2.9.1->stable-baselines3[extra]) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->tensorboard>=2.9.1->stable-baselines3[extra]) (1.3.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->stable-baselines3[extra]) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard>=2.9.1->stable-baselines3[extra]) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.13->stable-baselines3[extra]) (1.3.0)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard>=2.9.1->stable-baselines3[extra]) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard>=2.9.1->stable-baselines3[extra]) (3.2.2)
Building wheels for collected packages: AutoROM.accept-rom-license
  Building wheel for AutoROM.accept-rom-license (pyproject.toml) ... done
  Created wheel for AutoROM.accept-rom-license: filename=AutoROM.accept_rom_license-0.6.1-py3-none-any.whl size=446659 sha256=3d198e78ff0c60f13ff1edb657d4d28ad076e1879556e1a09a6ca6402181e249
  Stored in directory: /root/.cache/pip/wheels/6b/1b/ef/a43ff1a2f1736d5711faa1ba4c1f61be1131b8899e6a057811
Successfully built AutoROM.accept-rom-license
Installing collected packages: farama-notifications, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cud
  Attempting uninstall: ale-py
    Found existing installation: ale-py 0.7.5
    Uninstalling ale-py-0.7.5:
      Successfully uninstalled ale-py-0.7.5
Successfully installed AutoROM.accept-rom-license-0.6.1 ale-py-0.8.1 autorom-0.6.1 farama-notifications-0.0.4 gymnasium-0.29.1 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-1
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.3.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.14.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.11.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
```

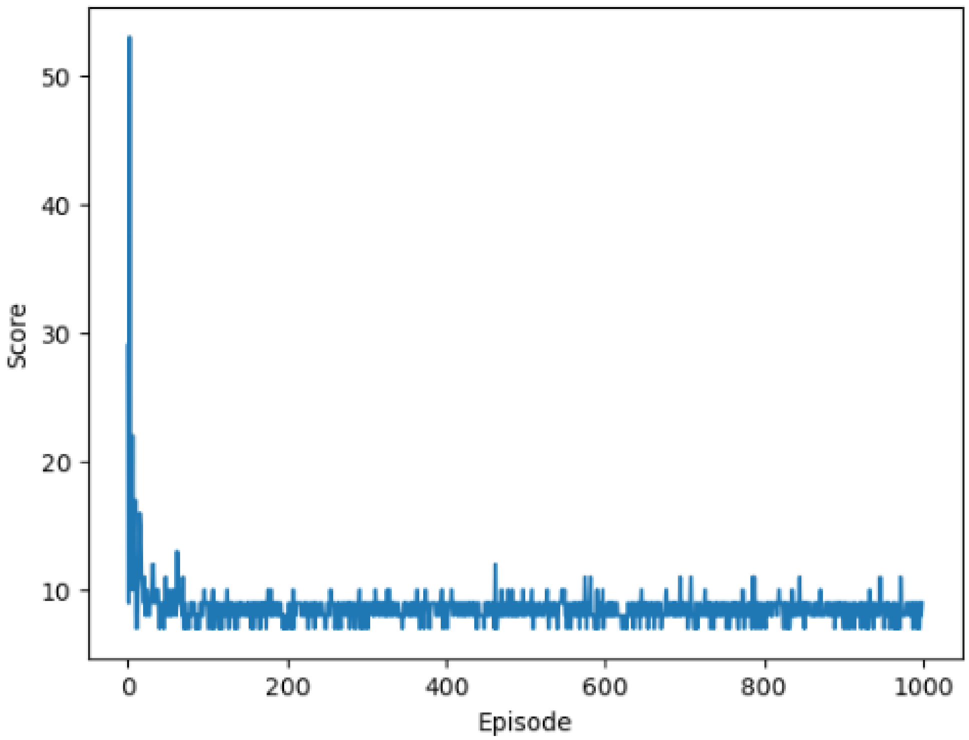
Шаг 2: Реализация DQN среды CartPole (полностью подключенная сетевая архитектура)



```
episode: 0/1000, score: 29, e: 1.0
episode: 1/1000, score: 9, e: 1.0
episode: 2/1000, score: 53, e: 0.86
episode: 3/1000, score: 16, e: 0.8
episode: 4/1000, score: 10, e: 0.76
episode: 5/1000, score: 13, e: 0.71
episode: 6/1000, score: 22, e: 0.64
episode: 7/1000, score: 10, e: 0.61
episode: 8/1000, score: 14, e: 0.56
episode: 9/1000, score: 17, e: 0.52
episode: 10/1000, score: 8, e: 0.5
episode: 11/1000, score: 7, e: 0.48
episode: 12/1000, score: 8, e: 0.46
episode: 13/1000, score: 10, e: 0.44
episode: 14/1000, score: 12, e: 0.41
episode: 15/1000, score: 16, e: 0.38
episode: 16/1000, score: 15, e: 0.35
episode: 17/1000, score: 11, e: 0.34
episode: 18/1000, score: 11, e: 0.32
episode: 19/1000, score: 10, e: 0.3
episode: 20/1000, score: 9, e: 0.29
episode: 21/1000, score: 11, e: 0.27
episode: 22/1000, score: 8, e: 0.26
episode: 23/1000, score: 9, e: 0.25
episode: 24/1000, score: 10, e: 0.24
episode: 25/1000, score: 9, e: 0.23
episode: 26/1000, score: 8, e: 0.22
episode: 27/1000, score: 8, e: 0.21
episode: 28/1000, score: 9, e: 0.2
episode: 29/1000, score: 9, e: 0.19
episode: 30/1000, score: 10, e: 0.18
episode: 31/1000, score: 12, e: 0.17
episode: 32/1000, score: 12, e: 0.16
episode: 33/1000, score: 9, e: 0.15
episode: 34/1000, score: 10, e: 0.15
episode: 35/1000, score: 9, e: 0.14
episode: 36/1000, score: 10, e: 0.13
episode: 37/1000, score: 10, e: 0.13
```



```
episode: 965/1000, score: 9, e: 0.01
episode: 966/1000, score: 7, e: 0.01
episode: 967/1000, score: 8, e: 0.01
episode: 968/1000, score: 7, e: 0.01
episode: 969/1000, score: 8, e: 0.01
episode: 970/1000, score: 9, e: 0.01
episode: 971/1000, score: 7, e: 0.01
episode: 972/1000, score: 11, e: 0.01
episode: 973/1000, score: 9, e: 0.01
episode: 974/1000, score: 8, e: 0.01
episode: 975/1000, score: 8, e: 0.01
episode: 976/1000, score: 8, e: 0.01
episode: 977/1000, score: 8, e: 0.01
episode: 978/1000, score: 8, e: 0.01
episode: 979/1000, score: 9, e: 0.01
episode: 980/1000, score: 9, e: 0.01
episode: 981/1000, score: 9, e: 0.01
episode: 982/1000, score: 9, e: 0.01
episode: 983/1000, score: 8, e: 0.01
episode: 984/1000, score: 8, e: 0.01
episode: 985/1000, score: 9, e: 0.01
episode: 986/1000, score: 9, e: 0.01
episode: 987/1000, score: 7, e: 0.01
episode: 988/1000, score: 9, e: 0.01
episode: 989/1000, score: 9, e: 0.01
episode: 990/1000, score: 9, e: 0.01
episode: 991/1000, score: 9, e: 0.01
episode: 992/1000, score: 7, e: 0.01
episode: 993/1000, score: 9, e: 0.01
episode: 994/1000, score: 9, e: 0.01
episode: 995/1000, score: 7, e: 0.01
episode: 996/1000, score: 7, e: 0.01
episode: 997/1000, score: 9, e: 0.01
episode: 998/1000, score: 8, e: 0.01
episode: 999/1000, score: 9, e: 0.01
```


50
40
30
20
10
Score
0
200
400
600
800
1000
Episode

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION (дата обращения: 13.02.2019)

[2] https://www.kaggle.com/datasets