Reinforcement Learning for Vision-based Robot Grasping

Project 15

Outline

- Introduction
- Method
- Experiments
- Result &Analysis
- QA

Introduction

RL in Vision-based Grasping

- **Challenges in vision-based robot grasping.**
  - Complex Visual Environments
  - Object Recognition and Localization
  - Depth Perception
  - Handling Unknown or Unseen Objects

- **The advantage of RL over traditional methods**
  - Adaptability to New Scenarios
  - Learning from Interaction
  - Handling High-Dimensional Data
  - Continuous Improvement
  - Decision-Making in Complex Environments

RL Approach vs. Traditional Methods in Robotic Grasping

Learning Methodology

Traditional Methods
- Often rely on deterministic algorithms.
- Require extensive programming and predefined rules
- Limited adaptability to changes in the environment or task.

RL Approach
- Employs a stochastic, trial-and-error learning process.
- Learns from interactions with the environment.
- Continuously adapts and optimizes strategies based on feedback.

Adaptability and Flexibility

Traditional Methods
- Perform well in structured, unchanging environments
- Struggle with variability and unpredicted scenarios

RL Approach
- Highly adaptable to new and changing environments
- Capable of generalizing learning to handle unseen situations

Handling Complex Data

Traditional Methods
- Limited in processing complex, high-dimensional data like real-time visual inputs
- May require simplification of the environment for effective operation

RL Approach
- Particularly effective in dealing with high-dimensional sensory data
- Leverages advanced neural networks for deep learning from complex inputs

Efficiency and Performance Over Time

Traditional Methods
- Consistent performance in known conditions.
- Performance does not improve over time unless reprogrammed.

RL Approach
- Performance improves with experience and data accumulation.
- Capable of optimizing strategies for efficiency and success rate.

Scalability and Generalization

Traditional Methods
- Scaling to different tasks can require significant reprogramming.
- Limited in generalizing learned behavior to different tasks.

RL Approach
- Better at scaling and transferring knowledge to different tasks.
- Can generalize strategies across various scenarios.

Objectives of RL-based Grasping

- **Enhancing Manipulation Skills**: Enhancing Robotic Grasping with Reinforcement Learning. Enhanced complexity, adaptability, and success in grasping tasks.
- **Learning and Adaptation**: Enable robots to learn grasping techniques incrementally and apply them to unfamiliar objects and situations .
- **Overcoming Programming Limitations**: Shift from hardcoded instructions to a model where robots learn autonomously from their environment.

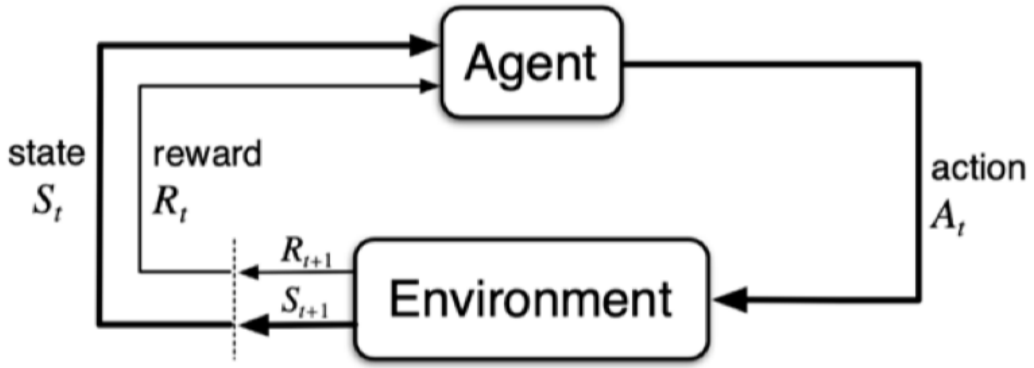
Method

We focus on exploring two of the prominent reinforcement learning algorithms:
the **Deep Q-Network (DQN)** and the **Branch Dueling Q-Network (BDQ)**

- **DQN**: Learn optimal policies directly from high-dimensional sensory inputs
- **BDQ:**
  - Splits neural network into two: one estimating the state value, the other estimating the advantage of each action.
  - Allows more nuanced decision-making
  - Helps the robot to distinguish between the value of the state and the value of the actions available
  - Crucial in grasping tasks: efficiency of actions can vary significantly based on the context.

Method

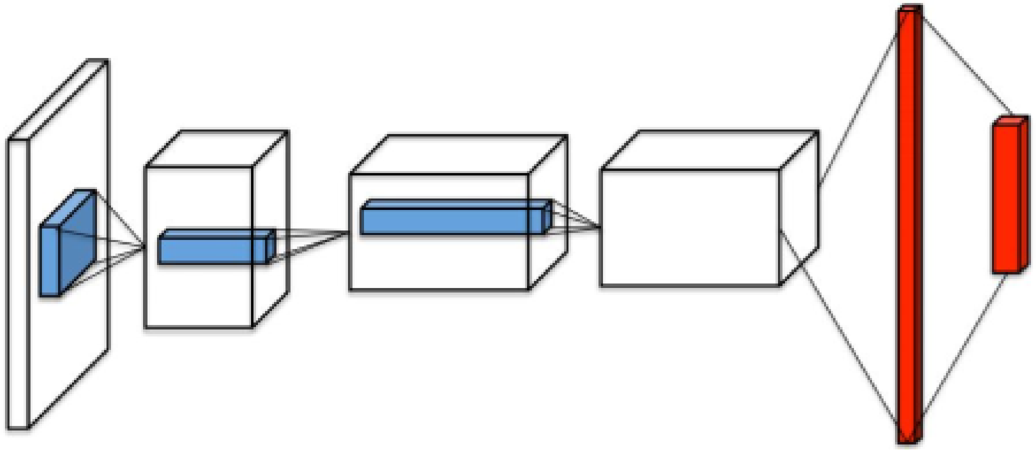
We focus on exploring two of the prominent reinforcement learning algorithms:
the **Deep Q-Network (DQN)** and the **Branch Dueling Q-Network (BDQ)**



Terminal Reward: $r_t = 10000$
Grasping Reward: $r_g = 100$
Δh Scale Coefficient: $c = 1000$
Time Penalty: $r_{tp} = 200$

- **State**: View of the environment from the eyes of the actor.
- **Reward**: $r = (\text{grasp detected}).(r_g + c.\Delta h) - r_{tp}$
- **Action:** the agent acts on the environment at time t, and the environment returns the reward and state of the action at time t + 1.

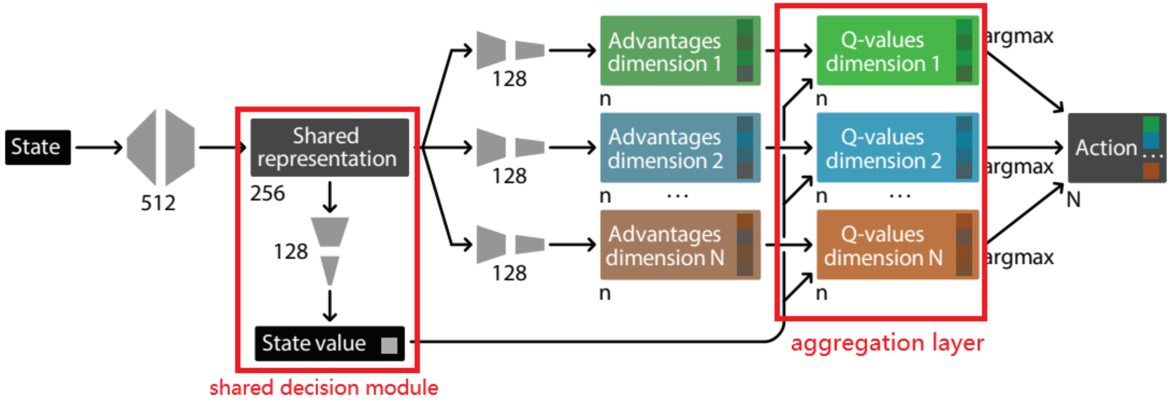
Method: Deep Q-Network (DQN) in Robotic Grasping



- **Basics of DQN**: A reinforcement learning algorithm that enables robots to make decisions based on environmental feedback.
- **Application in Grasping**: DQN helps robots analyze and learn from each interaction, progressively improving their ability to grasp and manipulate objects.

$$L(\theta) = E_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta^{-})) - Q(s,a;\theta)\right)^2\right]$$

Method: Branch Dueling Q-Network (BDQ)



- **Problem:** Both tabular Q-learning and DQN have been shown to suffer from the overestimation of the action values
- **BDQ:** a branching variant of the Dueling Double Deep Q-Network (Dueling DDQN)
  - **Shared Decision Module:** computes a latent representation, used for evaluation of the state value and the factorized (state-dependent) action advantages
  - **Aggregation Layer:** Combine the state value and factorized advantages to output the Q-value

$$L = \mathop{\mathbb{E}}_{(s,a,r,s')\sim D}\left[\frac{1}{N}\sum_{d}\left(y_d - Q_d(s,a_d)\right)^2\right]$$

A. Tavakoli, F. Pardo, and P. Kormushev. "Action branching architectures for deep reinforcement learning". In: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018 (2018), pp. 4131–4138. arXiv: 1711.08946.

Experiments

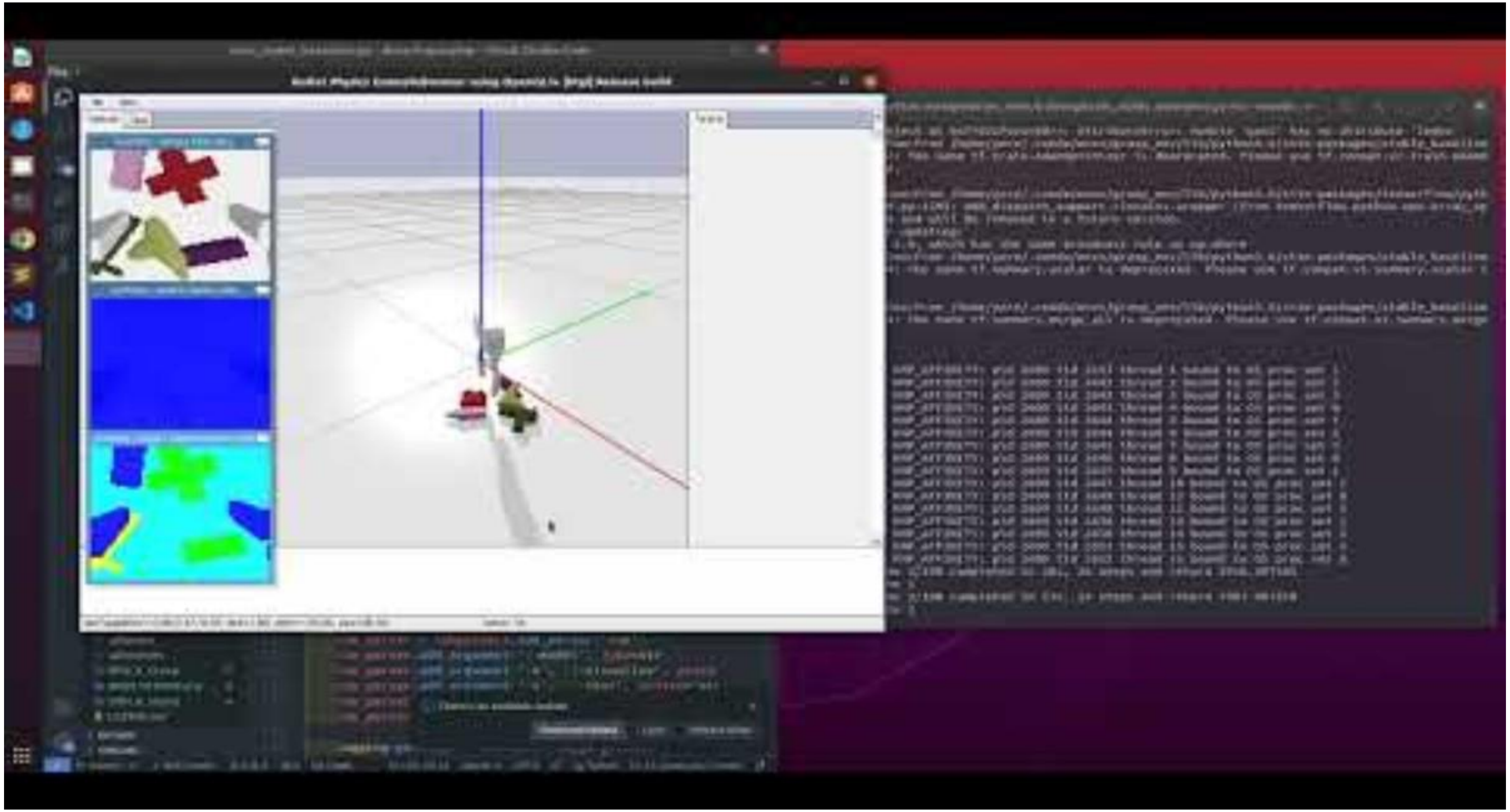
- Experimental setup
  - Dataset: pybullet-data
  - Robot Model: WSG50
  - Simulator: PyBullet
  - Framework: Gym
  - RL Library: stable-baselines
  - Evaluation Metrics: success-rate

BULLET PHYSICS LIBRARY

Gymnasium

Stable Baselines

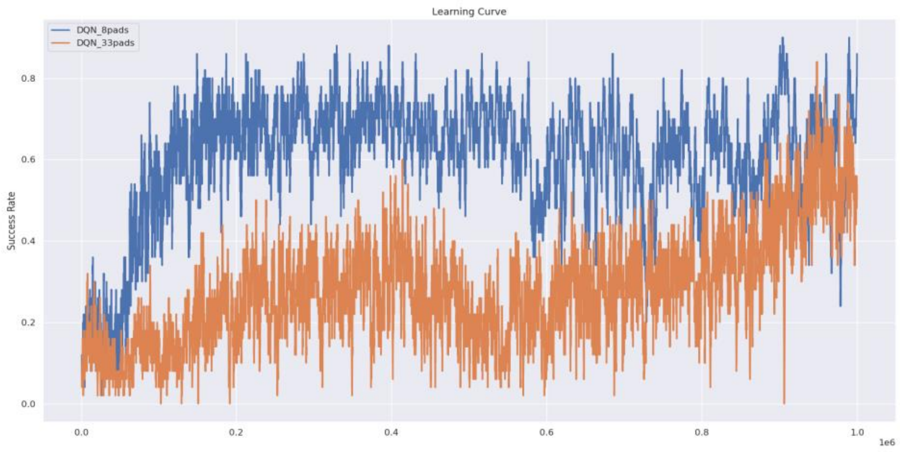
Experiments



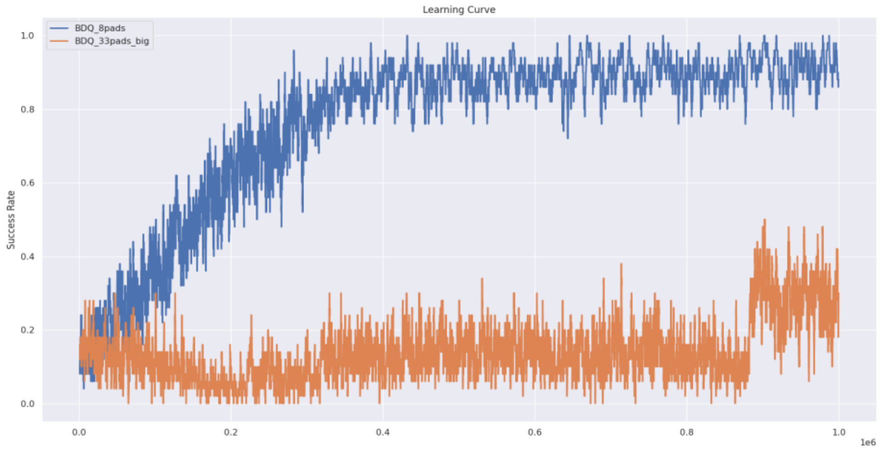
Results & Analysis

- **Goal:** how the neural network size affects the success rate
- **Method:** trained each model with varying action dimension padding 8 and 33
- **Conclusion:** model with smaller pads has a higher success rate on both training and testing set.
- **Analysis:**
    - **Exploration Challenges:** If the action space is larger, the agent may face more difficulty in effective exploration during the learning process.
    - **Curse of Dimensionality:** As the action space grows, the complexity of the problem may increase, making the learning process more challenging.

DQN:

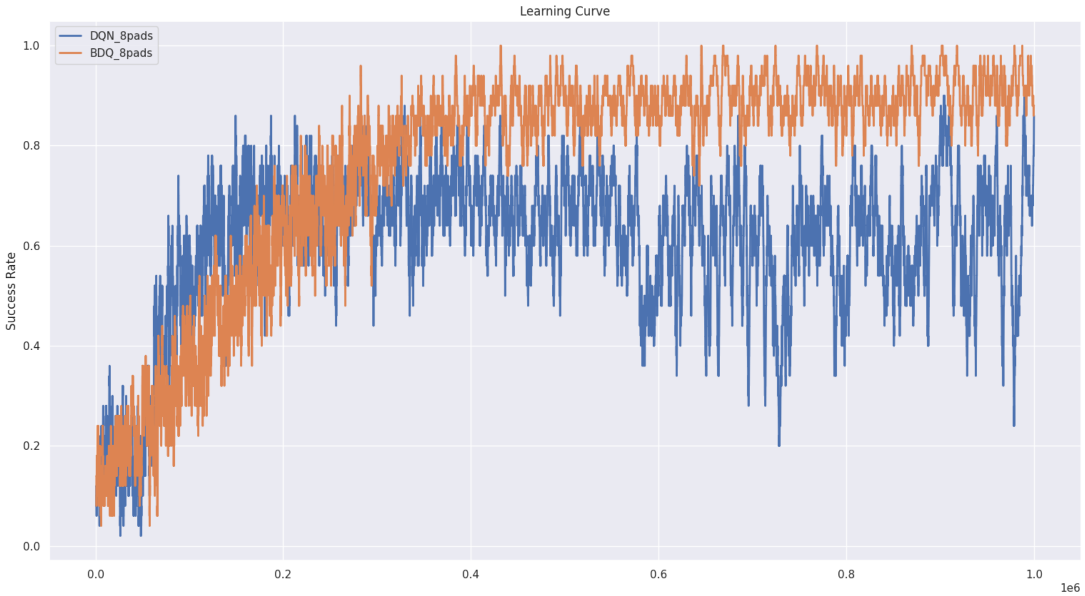


BDQ:



Models	Training Set (%)	Testing Set (%)
DQN_8pads	75%	65%
DQN_33pads	58%	49%
BDQ_8pads	91%	90%
BDQ_33pads	31%	26%

Results & Analysis



Models	Training Set (%)	Testing Set (%)
DQN_8pads	75%	65%
BDQ_8pads	91%	90%

DQN vs BDQ:

- Compare the best models of DQN and BDQ

Conclusion:

- BDQ: higher success rate
- DQN: converge faster

Further improvement:

- Transfer to a different scene to test the robustness.
- Parameter tuning:
  - Increasing the number of training episodes, adjusting learning rates

Thank you!

Q&A