KOÇ UNIVERSITY

College of Engineering

COMP 491 – Computer Engineering Design Project

Final Report

Deep Reinforcement Learning for Robotic Grasping

Participant information:
Arda Arslan 31760
Aykut Aykut 34039

Project Advisor(s)
Barış Akgün

May 29th, 2019

Table of Contents

I) Abstract

In industry, robots and automated machines are widely used to speed up processes and lower costs, and they are programmed according to mathematical and physical calculations. Each time a certain operation will be applied on a different material, all these calculations should be done once again and a new software is implemented from scratch. The main motivation is the automatization of this process to make it more efficient and generalizable for any industry. In this project, the main objective is to show that a certain operation can be done by a robot without doing new calculations and implementing a new software for different materials. For this purpose, a robotic arm is trained so that it becomes able to grasp and raise different shaped objects and the desired outcome of this project is a robotic arm with a trained artificially intelligent agent.

To simulate the actions of the robotic arm and to create an environment which includes the physics engine which handles collisions, gravity and frictional forces, ROS and Gazebo are used. The robotic arm is trained using a deep reinforcement learning algorithm called Deep-Q Learning with experience replay. The algorithm is implemented using Python. Tensorflow and Keras are used to implement the deep neural network structure. After trying different state spaces, action spaces and reward functions for the learning algorithm, an artificially intelligent agent which can grasp and raise different shaped objects is obtained. The weights of the deep neural network can be saved and these weights can be used to perform grasping and raising operations on different materials.

II) Introduction

One of the main problems in industry is equipment flexibility and durability. The equipment, especially robots, currently used in industry is programmed according to mathematical and physical equations and theories for kinematical movements which causes an increase in purchase costs, maintenance costs and waste of time when a new equipment is needed. This project will show that these costs can be decreased with an AI agent. The AI agent which controls the robotic arm will be able to learn how to grasp and raise different objects using Deep Q-Learning with experience replay algorithm. Since this AI agent is adaptable to the changes in its environment, it may also be used in space missions and any field in which human-beings have limited or no access due to the environment. The objective of this project is to have an artificially intelligent robotic arm which can grasp and raise different shaped objects. Grasping accuracy of the AI agent is measured and used to prove whether the project succeeds or not.

The main importance of this project is that if an agent with high grasping accuracy can be trained, this agent can be used for many purposes in different types of industrial factories. Today, if a robotic arm is needed to be used in a factory, it should be programmed so that it will become able to perform its task. So for each robotic arm, a different software should be designed. Instead of designing this software each time, one can also define the rewards, action space and the state space of the agent and train it. By this way the agent will again become able to perform its task eventually.

Today the manufacturing or processing speed of a product is crucial for industrial factories. There are still many parts of manufacturing and processing which are done manually. Having robotic arms that can perform tasks which are normally done manually, will definitely speed up manufacturing and processing in factories. This is one of the most significant aspect of this project, and since the factories are getting more automated each day, this also shows the timeliness of the project.

Three types of machine learning algorithms can be applied to the problem of grasping and raising: supervised, unsupervised and reinforcement learning algorithms. A literature review reveals that training an AI agent using a reinforcement learning algorithm rather than a supervised or an unsupervised algorithm yields higher accuracies in terms of this problem. The following methodologies are investigated and reviewed in detail:

a) *Supervised Learning Algorithms:* Saxena, Driemeyer and Ng [1] used 2-dimensional images of different objects and labeled the locations where the agent should perform grasping and they trained the agent using Logistic Regression. When the agent was asked to grasp an object which was similar to the ones in the training dataset, it had an accuracy of 90.0%. When it was asked to grasp an object which is novel, the agent had an accuracy of 87.8%. In addition to previous work, Rao et al. [2] used also the depth and visible light image data to improve the accuracy; however their agent had an accuracy of 87.5%. Mahler et al. [8] used a Convolutional Neural Network to train an agent which can predict the best location to grasp an object using depth images. They used a synthetic dataset named Dex-Net 2.0. This agent had a success rate of 99%. Pinto and Gupta [9] also used a Convolutional Neural Network to train an AI agent. The authors state that they used a training dataset which is 40 times larger than existing datasets. Their agent had an accuracy of 95.7% on seen objects and 76.9% on unseen objects.

b) *Unsupervised Learning Algorithms:* Ugur, Sahin and Oztop [3] let an agent interact with the environment and record the initial environment, performed action and the resulting impact.

They used Support Vector Machine Classifier to train the agent so that it can map each action to an impact. This agent had an accuracy of 90%.

c) *Reinforcement Learning Algorithms:* Stulp et al. [4] trained an AI agent which can learn to grasp an object despite the noisy sensor data. The agent learned to locate its hand near the object in such a way that when it closes its hand, the probability of a successful grasp will be highest. They used Dynamic Movement Primitives and Policy Improvement with Path Integrals. Their agent had an accuracy of 95% while trying to grasp T and H shaped objects. They also showed that the agent was more successful while grasping cylindrical objects. They state that their agent was successful in grasping objects which were never seen by the agent.

Quillen et al. [5] used different deep reinforcement learning algorithms to train an AI agent to grasp objects. These algorithms were Deep Deterministic Policy Gradient (DDPG), Deep Q-Learning (DQL), Monte Carlo (MC), Monte Carlo with Correction (C-MC), Path Consistency Learning (PCL) and Supervised Learning (S). They performed experiments for both on-policy and off-policy learning on datasets with sizes 10 thousand, 100 thousand and a million.

Off-Policy																	
Dataset Size: 10k						Dataset Size: 100k						Dataset Size: 1M					
DDPG	DQL	MC	C-MC	PCL	S	DDPG	DQL	MC	C-MC	PCL	S	DDPG	DQL	MC	C-MC	PCL	S
0.32	0.5	0.56	0.38	0.54	0.53	0.46	0.68	0.73	0.71	0.77	0.82	0.47	0.88	0.87	0.85	0.57	0.89

Table 1: Off-policy learning accuracies of Quillen et al.

On-Policy																	
Dataset Size: 10k						Dataset Size: 100k						Dataset Size: 1M					
DDPG	DQL	MC	C-MC	PCL	S	DDPG	DQL	MC	C-MC	PCL	S	DDPG	DQL	MC	C-MC	PCL	S
0.53	0.60	0.47	0.52	0.64	0.62	0.46	0.80	0.76	0.80	0.80	0.84	0.76	0.91	0.87	0.85	0.80	0.92

Table 2: On-policy learning accuracies of Quillen et al.

It is clear that increasing the dataset size will make a positive impact on the success rate of the AI agent in terms of grasping objects. In addition to this, it can be observed that for large datasets, Deep Q-Learning algorithm is the best off-policy reinforcement learning algorithm in terms of accuracy of grasping.

Gu et al. [6] used a deep and off-policy reinforcement learning algorithm which can open a door, pull a door, push a door, reach to an object, pick and place an object. The agent had an accuracy of 100%. They tried Deep Deterministic Policy Gradient algorithm, Normalized Advantage Function algorithm and Linear Normalized Advantage Function algorithm.

In his master's thesis, Rik [7] used Continuous Actor Critic Learning Automation to train an agent so that it can grasp objects. Since he used a continuous action space, he chose this algorithm(CACLA). He first applied a pre-learning so that the agent will only learn to locate itself near the object. After that he used only one neural network layer and he saw that the agent had an accuracy of 70%. After that he increased the number of network layers to 3 and observed that the agent had an accuracy of 100% while grasping the objects. He states that only the last layer deals with the finger position and the others deal with the hand position.

In this project, one of the reinforcement learning algorithms will be used since they yield the best accuracy in terms of grasping objects. During the literature review, we saw that CACLA was the best algorithm in terms of accuracy. However since the action space in this project will not be continuous, CACLA cannot be used. The second best off-policy reinforcement learning algorithm in terms of accuracy is Deep Q-Learning algorithm. Since it can be used with a discrete action space, in this project this algorithm will be used. The difference between our approach and the existing Deep Q-Learning algorithms will be the structure of layers and experience replay component. A custom structure for the layers of this Deep Q-Learning algorithm will be suggested and implemented. Experience replay will be used to avoid forgetting previous experiences and reduce correlations between experiences.

III) System Design

The design of this project consists of two main parts: design of the AI agent and design of the environment. To design the environment, ROS and Gazebo are used. ROS is an operating system for robots which helps developers build robot applications. Gazebo includes the physics engine which handles collisions, gravity and frictional forces. The actions taken by the robot and any change in the environment are simulated using Gazebo. The environment consists of an object, a table and a robotic arm and all of these are defined using a special marking language format named Unified Robot Description Format (URDF).

After inserting an object, the table and the robotic arm into the environment, the rules for the environment were implemented using Python. These rules include possible actions that the robotic arm can take, possible situations the robotic arm and the object can have and a custom function for rewarding the actions of the robotic arm.

The action space is discrete and consists of actions which increases or decreases angles of the joints of the robotic arm. The actions are combinations of + *action step size* and - *action step* *size* for joint angles. After trying different action step sizes, it is observed that 0.025 gives the best accuracy in terms of grasping.

Each state consists of two information: angles of the joints of the robotic arm and visual features of the object. These visual features are obtained using a library developed by the laboratory of project advisor. This library returns object features as a vector of size 308. To reduce its dimension, we used a deep neural network structure. The details of the deep neural network will be provided later in this section.

The reward function is as following: At each time step the robotic arm gets a reward of *-1*. The motivation is to make the robotic arm get closer to the object using the shortest path. In addition to this, at each time step the robot gets a reward of *10*(current_distance - next_distance)*. The motivation is to make the robot get closer to the object instead of moving away from the object. After trying different coefficients, *10* was chosen because it yielded the best grasping accuracy. If the fingers of the robotic arm gets significantly close to the object, the robotic arm gets a reward of *200*. If the robotic arm successfully grasps the object, it gets an additional reward of *200*. If the fingers of the robotic arm goes under the table, the robotic arm gets a reward of *-100*. If the center of gravity of the object goes under the table, the robotic arm gets a reward of *-100* again. If the robotic arm hits to the object so that the object moves significantly, more than *0.025*, then the robotic arm gets a reward of *-5*. If the robotic arm tries to go to a location which will cause a collision with the table, then it gets a reward of *-5*.

The Deep Q-Learning algorithm with experience replay is used to train the agent. At each time step, *20* randomly selected *(state, action, reward, state_next, terminal)* tuple from memory is updated according to the Bellman equation where discount factor of *0.95*, and the neural network is trained with the loss function of *MSE* and *Adam* as the optimizer with learning rate of *0.001*. ε-greedy approach is used to overcome the exploration-exploitation trade-off. Initially ε is set to *1.0* and ε-decay is applied with parameter *0.995*.

The agent initially uses a deep neural network with random weights while deciding on which actions to take. This causes the agent to act randomly at the beginning. The structure of the deep neural network is as following:

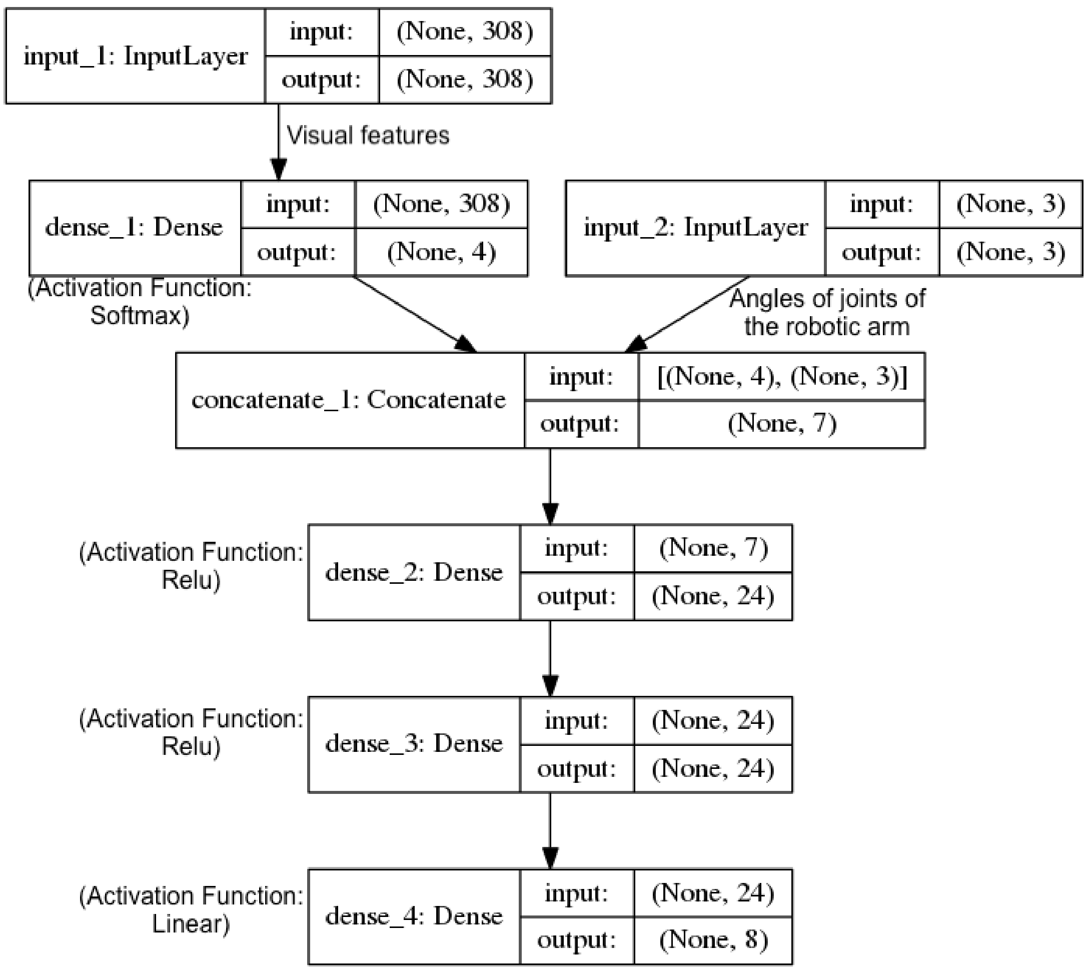

Figure 1: Deep Neural Network Structure

Input layer *input_1* represents the visual features of the object. Since we used 4 objects while training the robotic arm, we used a dense layer which encodes the visual features so that the new dimension becomes 4. Input layer *input_2* represents the joint angles of the robotic arm and these angles are represented with 3 dimensions. Output of the first dense layer *dense_1* and *input_2* are concatenated and after applying 3 dense layers with different activation functions, an output of dimension 8 is generated. Each node of this output layer *dense_4* corresponds to the prediction for the Q-value of an action that the robotic arm can take at its current state. The representation of the environment and the agent is as following:

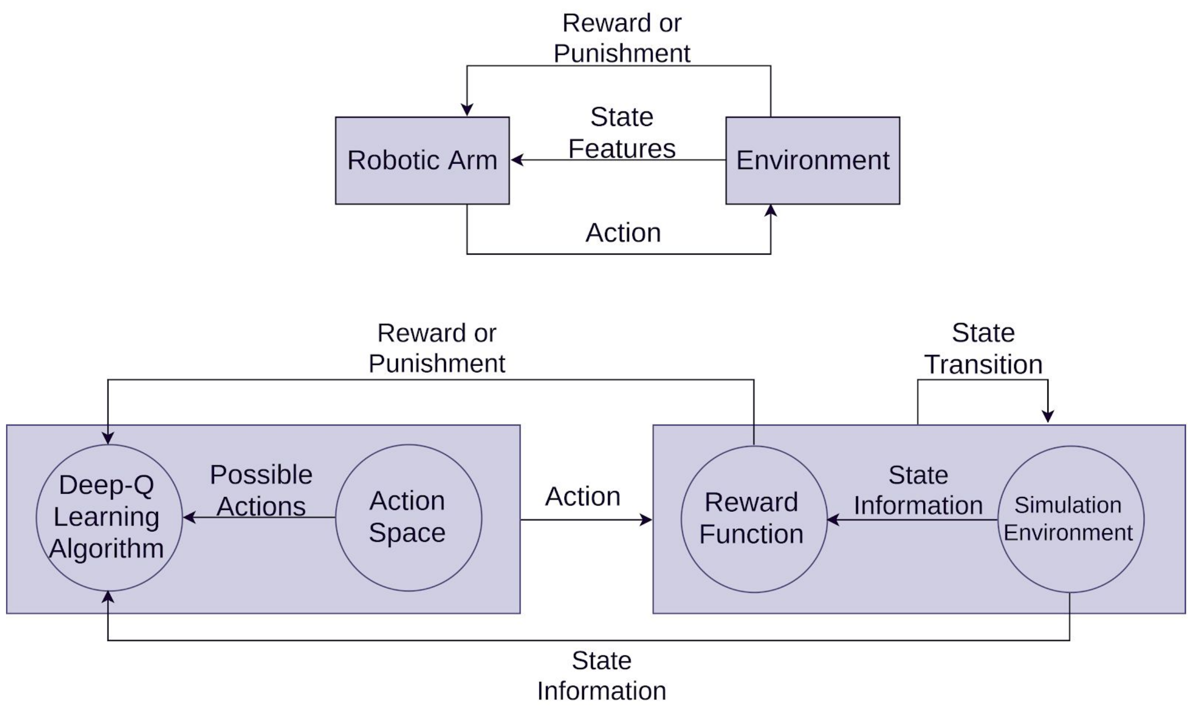

Figure 2: Representation of the environment and the agent

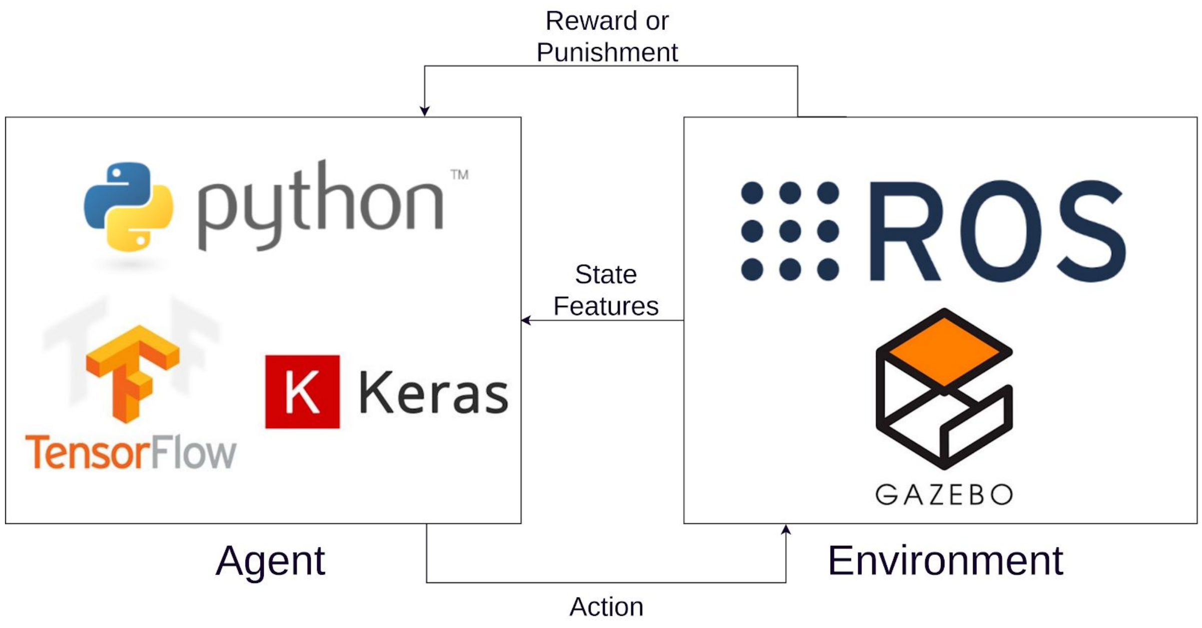

Figure 3: Representation of the environment and the agent including system architecture

The pseudocode [10] of the Deep-Q Learning with experience replay is as following:

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
**For** episode = 1, $M$ **do**
   Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
   **For** $t$ = 1,T **do**
      With probability $\varepsilon$ select a random action $a_t$
      otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
      Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
      Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
      Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
      Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
      Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
      Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the
      network parameters $\theta$
      Every $C$ steps reset $\hat{Q} = Q$
   **End For**
**End For**
```

Figure 4: Deep Q-Learning Algorithm Pseudocode

During designing the environment, the biggest problem encountered was simulating the action of grasping. The p, i and d values of the robotic arm controller were incorrect and this prevented the fingers of the robotic arm from moving. By the help of the assistant of the project advisor, these values were corrected and the robotic arm finally became able to perform the grasping action.

During designing the agent, the biggest problem encountered was finding the best hyperparameters for the model. Doing an exhaustive search would take too much time to find the optimal solution. So instead, the hyperparameters were tuned by trial and error strategy.

IV) Analysis and Results

At the beginning of the training phase, the robotic arm acts randomly since the weights of the deep neural network are randomly initialized; therefore, Q-value predictions are not sufficiently good. As the robotic arm is trained, it starts to take better actions for a given state by modifying the weights of its neural network. At each episode, a new object is spawned randomly in the simulation environment and the robotic arm is expected to grasp the object successfully. Total rewards for each episode when only one type of object is spawned can be found at Figure 5. Grasping success rate for each episode when only one type of object is spawned can be found at Figure 6.

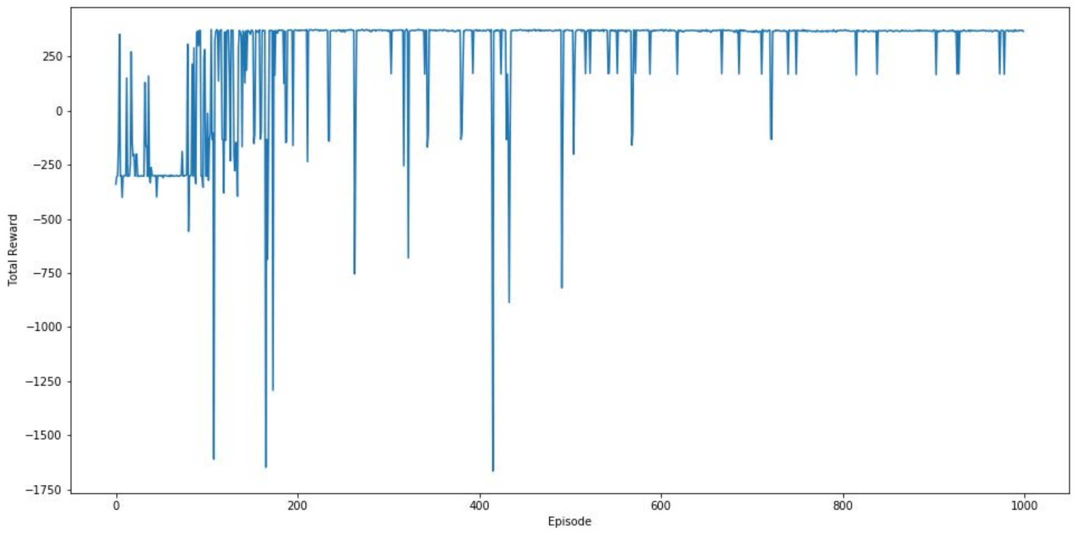

Figure 5: Total Reward vs. Episode Graph (Single Object)

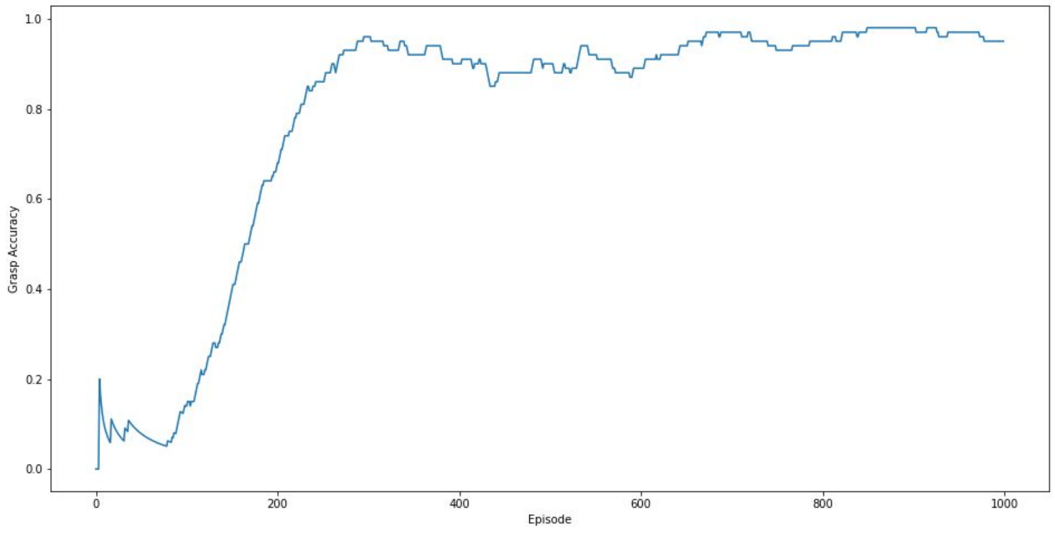

Figure 6: Grasping Success Rate vs. Episode Graph (Single object)

At it can be seen in Figure 5, the agent usually gets negative total rewards at the beginning of the training. However, after 500th episode, positive rewards are gained more and the agent starts to finish the episode with the maximum possible total reward.

In Figure 6, it can be observed that at the beginning of the training, grasping accuracy of the robotic arm is 0%. And it can also be seen that the robotic arm reaches a rate of 98% at the last episodes of the training. At the beginning, the robotic arm barely grasps the object until the 100th episode. Between 100th and 400th episodes, it is exploiting the model and managing to grasp successfully, and thus, increases the rate. After the 400th episode, the robotic arm has an average grasp rate of 93% with a maximum of 98%. The robotic arm is successfully trained using Deep Q-Learning with experience replay and a custom deep neural network structure so that it becomes able to grasp objects in the simulation environment.

When four types of objects are spawned randomly, the model should be trained much longer. When the model is trained for 1750 epochs, its grasping accuracy is 43%. However it is observed that the accuracy of grasping increases as the training continues. One should train the robotic arm for approximately 10000 epochs to see a high accuracy while four types of objects are spawned randomly. Grasping accuracy of the robotic arm when four objects are spawned randomly can be found at Figure 7. As it can be seen in Figure 7, grasping success rate has a positive trend. If the model is trained for more epochs, it would yield higher accuracy.

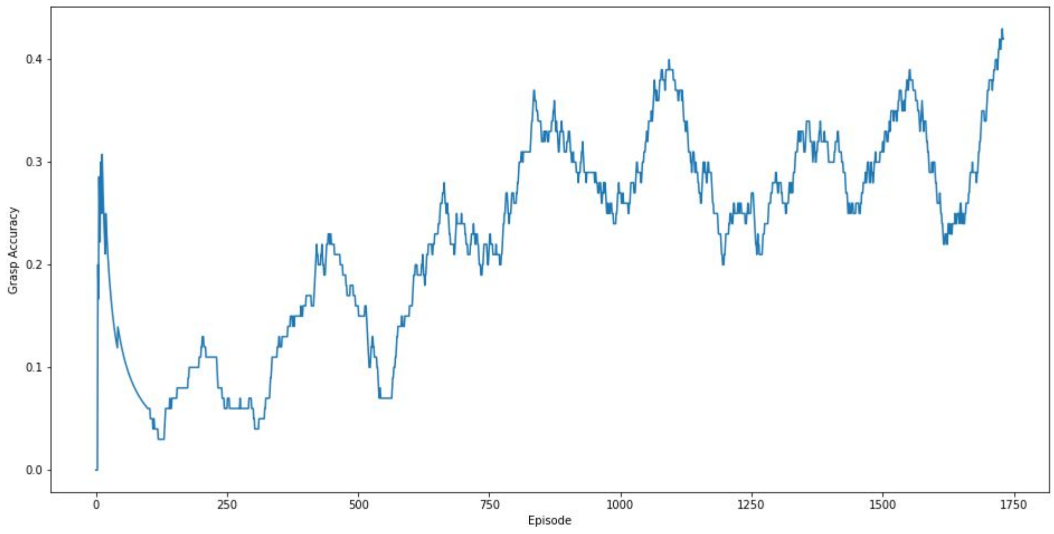

Figure 7: Grasping Success Rate vs. Episode Graph (Four objects)

V) Conclusion

In this senior design project, Deep Q-Learning algorithm with experience replay is implemented and a new deep neural network structure is suggested. It is observed that the Deep Q-Learning algorithm along with the neural network model accomplished the task of grasping with 93% accuracy when one type of object is spawned. The robotic arm is trained so that it becames able to grasp and raise four different randomly chosen objects, a box, a cylinder, a sphere and a rabbit as an instance for complex objects in the simulation environment. The trained robotic arm is capable of grasping and raising objects using the shortest path to the object. A robotic arm with an AI agent which is trained using reinforcement learning decreases the production cost and time, therefore, it is beneficial for any industry. As a future work, the robotic arm can be trained using other reinforcement algorithms, especially on-policy algorithms, for the same task of grasping for benchmark and the object set can be enlarged.

VI) References

[1] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic Grasping of Novel Objects using Vision,” The International Journal of Robotics Research, vol. 27, no. 2, pp. 157–173, Feb. 2008.

[2] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, “Grasping novel objects with depth segmentation,” in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[3] E. Ugur, E. Sahin, and E. Oztop, “Unsupervised learning of object affordances for planning in a mobile manipulation platform,” in 2011 IEEE International Conference on Robotics and Automation, 2011.

[4] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, “Learning to grasp under uncertainty,” in 2011 IEEE International Conference on Robotics and Automation, 2011.

[5] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, “Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods.” in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018.

[6] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017.

[7] T. Rik, “Learning to Grasp Objects with Reinforcement Learning,” Master’s Thesis, 2018.

[8] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”, in Robotics: Science and Systems (RSS), 2017.

[9] L. Pinto, and A. Gupta, “Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours”, CoRR, abs/1509.06825, 2015.

[10] Jonathan Hui. 2018. RL — DQN Deep Q-network. Retrieved May 14, 2019 from

https://medium.com/@jonathan_hui/rl-dqn-deep-q-network-e207751f7ae4

VII) Appendix

Source code can be found at github: https://github.com/ardarslan/robograsp/