

Deep Reinforcement Learning For A Four Degree Of Freedom Robot Arm Accelerated By Human Demonstration

Kirk Duran
College of Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
kduran02@calpoly.edu

Roel J. Hernandez
Intelligent Systems Lab
Sonoma State University
Rohnert Park, CA, USA
roeljoseh@gmail.com

Sudhir Shrestha
Intelligent Systems Lab
Sonoma State University
Rohnert Park, CA, USA
shrestsu@sonoma.edu

Abstract— This work investigates the use of deep reinforcement learning (DRL) for training a four-degree-of-freedom (4-DOF) robotic arm to efficiently and adaptively reach arbitrary target positions. A dueling double deep Q-network (D3QN) manages the large state-action space, while human demonstration data guides the agent's policy toward more human-like and effective trajectories. Experimental results show that integrating human demonstrations enables the robot arm to achieve more precise target positioning more frequently and in fewer steps than a purely self-trained agent. To assess generality and adaptability, the approach is tested on both a PhantomX Reactor robot arm and a Dobot Magician robot arm, with adjustments made to the Denavit-Hartenberg parameters of the latter. After running 5000 simulated training episodes on each platform, the resulting model is then implemented on the Dobot hardware for 50 episodes, demonstrating the feasibility of accurately controlling a robotic arm model different from the one used during training. The method's potential extends to practical computer vision-guided tasks such as pick-and-place or peg insertion, thus showcasing its versatility in both simulation and physical implementations.

Keywords— *Deep Reinforcement Learning, Artificial Intelligence, Robotic Arm, Human Demonstration*

I. INTRODUCTION

Robotic manipulators are increasingly employed in tasks spanning manufacturing, medical, and agricultural applications, often requiring adaptation to unstructured environments [1], [2]. Traditional approaches rely on explicit programming and stable environments, limiting adaptability to variability in objects, configurations, or conditions. Reinforcement Learning (RL) offers an alternative: robots can learn policies through trial and error without requiring fully specified models [3],[4],[5],[6]. Yet, high-dimensional state-action spaces in complex manipulators increase the data and time required for learning [6].

Deep Reinforcement Learning (DRL) extends RL by approximating value functions with deep neural networks, enabling learning in continuous or large discrete state-action spaces [7],[8],[9]. While DRL has facilitated learning policies for complex tasks, purely self-exploratory methods may still require extensive sampling [1],[3]. To mitigate this, researchers have integrated human demonstrations to guide exploration and accelerate learning, especially in tasks where designing a reward function or exploring large state-action spaces is challenging [10],[11],[12]. The use of deep Q network, artificial neural network, and artificial deep Q network through demonstration-based methodologies are presented in [13].

Prior studies have often validated RL in simulation or for low-degree-of-freedom tasks [3],[4],[8]. Although some works have achieved compelling results in real-world setups [5],[7], they typically rely on large datasets, multiple robots, or are restricted to simpler motions or one-dimensional tasks [3],[10],[11]. Incorporating human demonstrations helps bootstrap the learning process, reducing the burden of random exploration [10],[11],[12], but further research is needed for higher-dimensional arms and real-world scenarios.

Here, we train a 4-DOF PhantomX Reactor robot arm [23] using a dueling double deep Q-network (D3QN) [21][6] to learn a reaching task. By discretizing joint actions and integrating human demonstration data, we improve learning speed and final accuracy over a purely exploration-based approach. Our results show that the human demonstration pre-trained agent can reach closer to the target more frequently and with fewer steps. We also compare results to a parallel implementation on a Dobot Magician arm, illustrating the method's adaptability. Future extensions include vision-based position estimation [1],[5] and more complex reward functions [6],[8].

II. THEORETICAL ANALYSIS

A. Deep Reinforcement Learning Background

RL can be formalized as a Markov Decision Process (MDP), where an agent observes states, selects actions, and receives rewards. Q-learning [14],[15],[16] learns an action-value function that estimates long-term rewards, but tabular Q-learning becomes infeasible in large state-action spaces. DQNs use neural networks to approximate Q-values, enabling learning in high-dimensional problems [8],[9]. Double DQN reduces overestimation bias by separating action selection and value evaluation [8]. Dueling DQNs introduce separate estimators for state-value and advantage, improving policy evaluation [17], [18]. Combining these approaches yields a D3QN that provides stability and efficiency in large discrete action spaces.

B. Robot Arm, DH Parameters, and Kinematics

We use a 4-DOF PhantomX Reactor robot arm [19]. Each joint is revolute, and the arm configuration is described by Denavit-Hartenberg (DH) parameters to compute forward kinematics [20], [21]. By specifying link lengths and joint angles ($\theta_1, \theta_2, \theta_3, \theta_4$), we compute the end-effector position (x, y, z). Although forward kinematics can introduce modeling errors, it is sufficient for our reaching task and for evaluating the learned policies.

III. STUDY DESIGN

A. Experimental Setup

The robot arm is controlled via Python using a microcontroller and Dynamixel AX-12 servo motors. Positions are read and written in degrees, with approximately 0.3° resolution. We restrict actions to $\pm 1^\circ$, 0° increments per joint, yielding 81 possible actions (3^4) [19], [22]. The reward function assigns +1 if the chosen action reduces the distance to the target, and -1 otherwise. Episodes terminate if a certain reward threshold is reached, too many bad moves occur consecutively, or the agent reaches the target within a defined epsilon radius.

A replay memory stores transitions (state, action, reward, next state, done), sampled in batches to train the D3QN. The network uses ELU activations [18], the adam optimizer [23], and a huber loss function [24], trained with a learning rate of 0.001 and batch size of 64.

B. Simulation

Though not the primary focus, a simulated environment can validate initial parameters and reward structures. Preliminary simulation testing helps ensure that the discretization strategy, network architecture, and reward function work as intended before deploying on hardware

C. Hardware Implementation (PhantomX Reactor)

For hardware experiments, the PhantomX Reactor is placed in a workspace with randomly generated starting and target positions. The robot executes random exploration to build an initial dataset. A D3QN, with a 7-dimensional input (x, y, z of target and $\theta_1, \theta_2, \theta_3, \theta_4$ of the arm), and 81 outputs (one per action), is trained. We use Exponential Linear Unit (ELU) activations, the Adam optimizer, and the Huber loss function for stable training.

D. Integration of Human Demonstrations

Human demonstration data is collected by disabling torque and physically guiding the robot arm from a starting position to a target [11], [12]. Each recorded trajectory is discretized into 1° steps and assigned a +1 reward. Though this introduces some noise, it accelerates the learning process by providing structured, near-optimal examples for the agent to mimic. This approach reduces the exploration burden and leads to faster convergence and more human-like movement patterns.

IV. RESULTS AND DISCUSSION

A. Simulation

Initial simulation tests (if employed) would show that the D3QN setup successfully learns stable policies in a simplified environment. The agent begins with random performance and gradually increases cumulative rewards per episode, confirming that the selected architecture and hyperparameters are suitable for the real-world task.

B. Hardware Implementation in PhantomX Reactor

Without human demonstrations, the agent learned to reach within 5 cm of targets 48.2% of the time and within 1 cm 11.2% of the time. Incorporating human demonstration data improved performance substantially: the agent reached within 5 cm 92.7% of the time, though its 1 cm accuracy was 4.4%.

The demonstrations accelerated training and guided the policy toward more linear, efficient trajectories, reducing the number of steps required to reach close proximity.

These results highlight the benefits of using human demonstrations as a prior. However, inaccuracies in forward kinematics and limited demonstration datasets introduced errors that prevented perfect fine-tuning near the target.

We also use a Dobot Magician Robot Arm [25]. Each joint is a revolute joint creating an R,R,R,R - waist, shoulder, elbow, and wrist configuration. In polar coordinates, the waist axis controls the

θ position and the remaining 3 axes control the radius r and height h . The limits of each axis are: $\theta_1 = 180^\circ$, $\theta_2 = 85^\circ$, $\theta_3 = 100^\circ$, $\theta_4 = 90^\circ$.

C. Results

We ran the simulation for the trained model 5,000 episodes (the same number of episodes used to train the model) to evaluate the accuracy of the trained model. We ran the simulation with two different kind of 4 DOF Robotic Arm. One, the ArbotiX Robot Arm used during the model trained phase, the other one, Dobot Magician Robot Arm. During the training phase of the model, the agent reached in 92% of the cases a distance of less than 5 cm, therefore we can take as the range of acceptance of the simulation episodes, distances less than 5 cm.

Table I shows some statistics of the simulations performed with both robot. For example, we can observe that the mean value of the distance error and the Euclidean distance are within the range of acceptance, 3.051 cm and 3.520 cm respectively for Arbotix Robot Arm. The distance error and the Euclidean distance are 6.155 cm and 5.723 cm respectively for Dobot Magician Robot Arm. Calculating the standard deviations shows that for the distance error there is a dispersion of 2.632 cm with respect to the mean and for the Euclidean distance of 1.050 cm, which indicates that there is a good behavior of the simulation in reaching or approaching the target point for the Arbotix Robot Arm. For the Dobot Magician Robot Arm the standard deviations shows that for the distance error there is a dispersion of 1.759 cm with respect to the mean and for the Euclidean distance of 1.595 cm. The rmse value for Arbotix Robot Arm confirms that there is a good behavior of the simulation, since it is 2.121 cm. The rmse value for the Dobot magician Robot Arm is 3.430 cm.

TABLE I. DISTANCE ERROR, ECUCLIDIAN DISTANCE, AND REWARD

Metric	Robot Type	
	ArbotiX	Dobot
Average Distance Error	3.051 cm	3.051 cm
Average Euclidean Distance	6.155 cm	6.155 cm
Average Reward	3.520 cm	3.520 cm

Table II shows the number of episodes and percentages that fall within the acceptance regions. For example, for ArbotiX Robot Arm, we can observe that the percentage of episodes with distance error less than 1 cm is 18.32% of the simulations and less than 5 cm is 83.96%, which again shows

a good behavior of the robot. Regarding the distance error and reward we can see that in this case for distance error < 5 cm and reward < 200 there are 3,085 episodes, which represents 61.70% of all simulations lying within the acceptance margins, so we can mention that the behavior of the simulation is satisfactory for this Robot. For Dobot Magician Robot arm, we can observe that there is not of episodes with distance error less than 1 cm and with distance error less than 5 cm is 27.48%, which is a low percentage this Robot. Regarding the distance error and reward we can see that in this case for distance error < 5 cm and reward < 200 there are 1,374 episodes, which represents 27.48% of all simulations lying within the acceptance margins, so we can mention that the behavior of the simulation is not satisfactory for Dobot Magician Robot Arm.

TABLE II. SIMULATION RESULTS SUMMARY

Metric	ArbotiX		Dobot	
	Value	Percent	Value	Percent
Numbers of Episodes with Distance Error < 1 cm	916	18.32%	0	0.00%
Numbers of Episodes with Distance Error < 5 cm	4,198	83.96%	1,374	27.48%
Distance Error < 5 cm and Rewards < 100	1,505	30.10%	1,350	27.00%
Distance Error < 5 cm and Rewards < 200	3,085	61.70%	1,374	27.48%
Distance Error < 1 cm and Rewards < 100	203	4.06%	0	0.00%
Distance Error < 1 cm and Rewards < 200	635	12.70%	0	0.00%

Figure 1 show the comparative distance error base on the equation (1) for both Robot arm. The Arbotix robot agent reached a minimum position error of 0.75 cm in 207 steps with a reward of 174. The Dobot Magician robot agent reached a minimum position error of 1.7 cm in 78 steps with a reward of 59.

$$error = r \times |\Delta\theta| + \sqrt{\Delta r^2 + \Delta h^2} \quad (1)$$

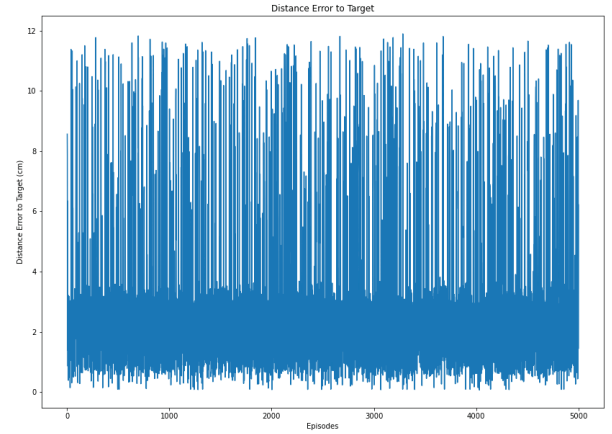
Figure 2 show the comparative Euclidean distance, equation (2), for both Robot arm. The Arbotix robot agent reached a minimum Euclidean distance of 0.45 cm in 81 steps with a reward of 60. The Dobot Magician robot agent reached a minimum Euclidean distance of 1.7 cm in 815 steps with a reward of 66.

$$d(P_t, P_p) = \sqrt{(X_t - X_p)^2 + (Y_t - Y_p)^2 + ((Z_t - Z_p)^2)} \quad (2)$$

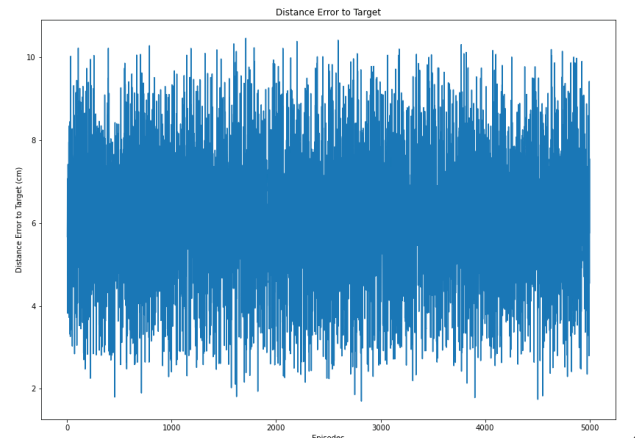
In figure 3 show the comparative Rewards obtained during the simulation for the robot agent with the trained model. For ArbotiX Robot Arm we can see 128 episode with a minimum reward of -50 (Stop condition of Algorithm "rewards < -50 ") and 148 episodes with the maximum reward of 300 (Stop condition of Algorithm "reward > 300 "). For the Dobot Magician Robot Arm we can see 1,311 episodes with a reward of -10 (Stop condition of Algorithm, 10 consecutive negative rewards") and 2 episodes with a reward of 152, none of the episodes for Dobot Magician Robot Arm reached the maximum reward of 300.

As we can see from the simulation results, ArbotiX Robot Arm performed better than Dobot Magician robot Arm, this is due to:

- The model was trained using ArbotiX Robot Arm parameters (link length, θ_i angle range).
- The human demonstration data to train the Neural Network was obtained based on ArbotiX Robot Arm movement position.
- ArbotiX Robot Arm has a higher range of motion in $\theta_1(0 \sim 300^\circ)$ compared to Dobot Magician Robot Arm $1(-90^\circ \sim 90^\circ)$.



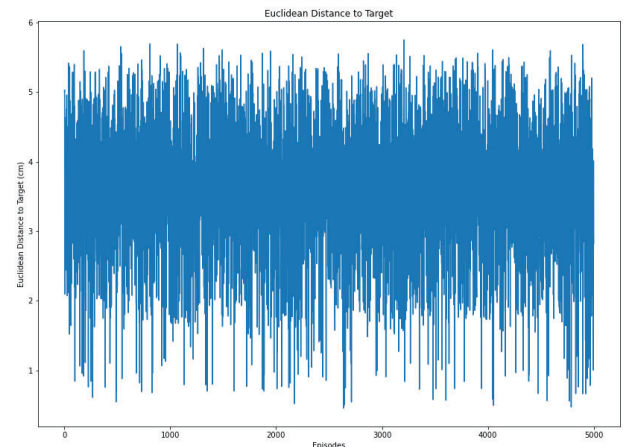
(a) ArbotiX



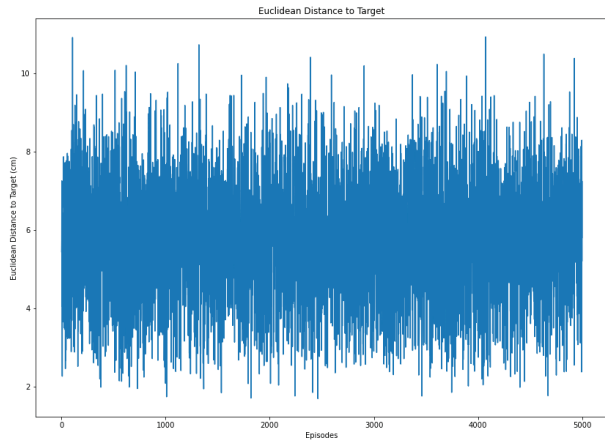
Dobot

(b)

Fig. 1. Distance error for a 4-DOF robot agent simulation results for 5,000 episodes at 300 steps with human trained model.



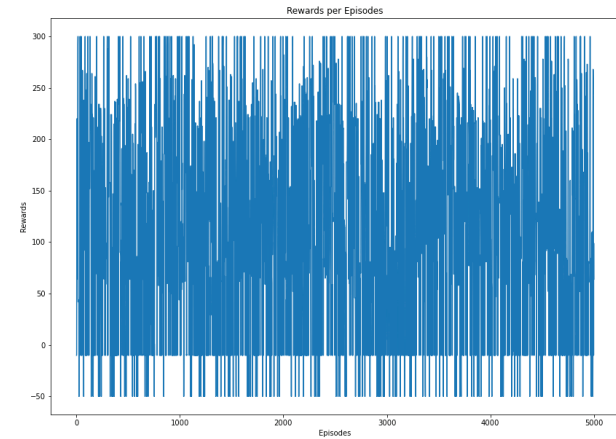
(a) ArbotiX



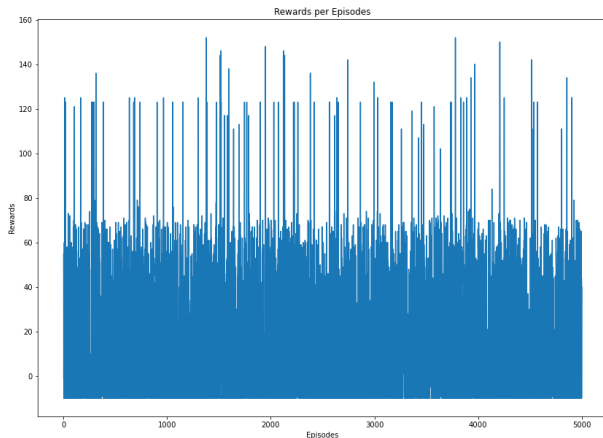
(b) Dobot

Fig. 2. Distance error for a 4-DOF robot agent simulation results for 5,000 episodes at 300 steps with human trained model.

After running the trained model 50 times using the Dobot Magician Robotic arm we got the results shown in Table III. Table III shows some statistics of the execution performed with the robot Dobot magician. For example, we can observe that the mean value of the distance error and the Euclidean distance are within the range of 6.796 cm and 6.164 cm respectively. Calculating the standard deviations shows that for the distance error there is a dispersion of 1.323 cm with respect to the mean



(a) ArboitX



(b) Dobot

Fig. 3. Rewards for a 4-DOF robot agent simulation results for 5000 episodes at 300 steps with human trained model.

TABLE III. DOBOT MAGICIAN RESULTS

Metric	Results
Average Distance Error	6.796 cm
Average Euclidean Distance	6.164 cm

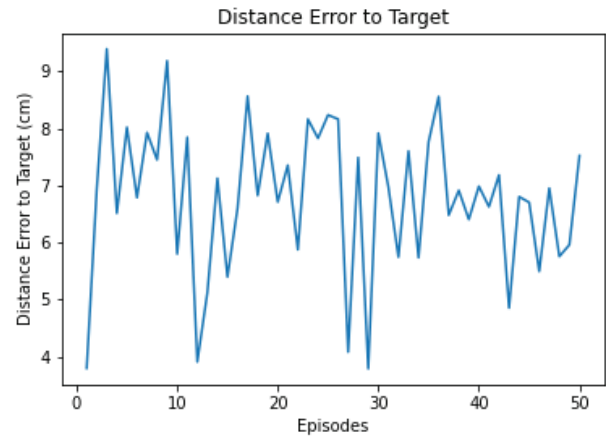
and for the Euclidean distance of 1.312 cm, which indicates that there is room for improvement

The results obtained with the trained model implemented in Dobot Magician Robot Arm are similar to those obtained in the simulation using the Dobot Magician Robot Agent, there was no episode where a distance of less than 1 cm was reached, and the percentage of episodes with a distance of less than 5 cm is less than 50% of the cases, 10% for the Dobot Magician Robot Arm implementation and 27% in the Dobot Magician Robot Agent simulation. Table IV shows the number of episodes and percentages that fall within the acceptance regions. For example, we can observe that the percentage of episodes with distance error less than 5 cm is only 10%.

TABLE IV. DOBOT MAGICIAN RESULTS

Metric	Results	
	Value	Percent
Numbers of Episodes with Distance Error < 1 cm	0	0.00%
Numbers of Episodes with Distance Error < 5 cm	5	10.00%
Distance Error < 5 cm & Step < 100	5	10.00%
Distance Error < 5 cm & Step < 200	5	10.00%
Distance Error < 1 cm & Step < 100	0	0.00%
Distance Error < 1 cm & Step < 200	0	0.00%

Figure 4 show the distance error base on equation 3.2 and Euclidean distance 3.1, the Robot Arm reached a minimum position error of 3.7 cm in 47 steps. The minimum Euclidean distance reached was 3.6 cm in 47 steps.



(a) Distance Error

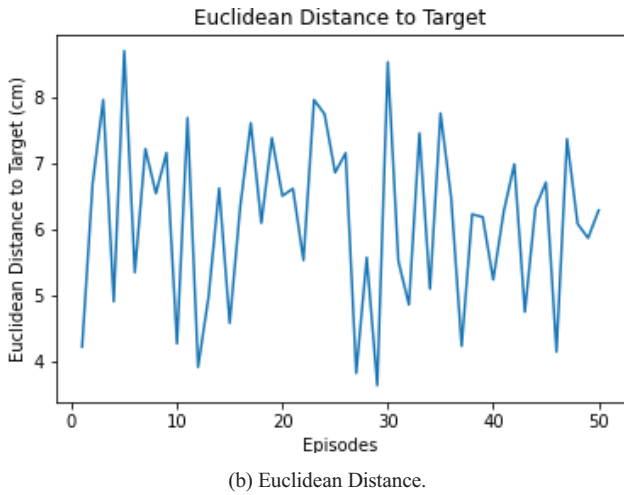


Fig. 4. Distance error and Euclidean Distance for a 4-DOF robot agent dobot results for 50 episodes at 300 steps with human trained model.

During the simulation, the agent using the PhantomX Robot Arm reached on average 3.051 cm from the target, while with Dobot magician Robot Arm it reached on average 6.155 cm, also with PhantomX Robot Arm, 83% of the episodes reached a distance of less than 5 cm and in 18% a distance of less than 1 cm. With Dobot Magician Robot Arm, 27% of the episodes reached a distance of less than 5 cm, but none of episode reached a distance less than 1 cm. A similar behavior was observed when implemented on the Dobot Magician Robot Arm, the Robot never reaching less than 1 cm from the target.

D. Discussion

From the simulation result between the two Robot Arm and the execution performed with the robot Dobot magician we can see a better performance with the Arbotix Robot arm. The difference between the simulation and the implementation in the robot are due to working space in each robot are different. The Arbotix Robot has limits of each axis: 1 = 300°, 2 = 180°, 3 = 200°, 4 = 210°, while the Dobot magician has limits of each axis: 1 = 180°, 2 = 85°, 3 = 100°, 4 = 90°. Also, the model was trained with the parameter for ArbotiX Robot Arm and its Human Demonstration dataset.

Other works done before has similar results as we get in our project, i.e. [3] using G-learning and running 50 episodes with $\epsilon = 0.1$ has a successful rate of 64%, while [26] With QT-Opt Algorithm with 5,000 real-world data reaches a success rate of 91%. Similar to the rate reaches by the Simulation with ArbotiX Robot agent of 84%.

The Dobot Magician Robot Agent reached a successful rate of 27%. To increase the accuracy and overcome the challenges of training robot with deep reinforcement learning we can use generalization, specifically data diversity as state in [26] "Good data diversity that covers the space of generalization we care about is critical to have good performance with deep learning." We can train the model using a dataset that combined the human demonstration position for different Robot Arm and then implement the trained model with each robot to compare the behavior.

V. CONCLUSIONS

We demonstrated that a 4-DOF robot arm could be trained via DRL to reach target positions with a D3QN architecture,

and that human demonstration data significantly accelerates training and improves accuracy. The human-trained agent not only learned faster but also produced more human-like and efficient motions, often reaching near-target positions in fewer steps.

Nevertheless, forward kinematic inaccuracies, limited demonstration data, and the complexity of the workspace restricted fine-grained positioning. Future work will consider model-free methods using external sensors [1], [5], more extensive demonstration datasets, improved reward functions [6], [8], and possibly incorporating natural language interfaces for more intuitive human-robot interaction.

When integrated with computer vision systems and extended to pick-and-place or peg insertion tasks, such DRL-based approaches hold promise for deploying flexible and adaptive robotic solutions in real-world environments.

We were able to implement the trained model on hardware (Raspberry Pi) and control the robotic arm via the interface in Python. In this project it was possible to implement in hardware the trained model and control The Dobot Magician Robot Arm using python on Raspeberry Pi. The first thing we did was the simulation of the model using the Agent and libraries of the original robot (Arbotix Robot Arm used to train the model). To perform the simulation we adjusted the Python program so that it could run on Raspberry Pi with Python 3 [27]. Once the simulation with the original Robot was finished we obtained the statistical data of the simulation, showing a similar behavior to that achieved during the training phase. Then we proceeded to make adjustments in the Python code to adjust the program to Dobot Magician Robot Arm's parameters as well as the library to interact with Dobot Magician. Using Python and Raspberry Pi we were able to implement in Dobot Magician Robot Arm the trained model. The agent was able to control the robot and move it relatively close to the target position, even though it did not achieve in most cases values within the acceptable range.

During the simulation for 5,000 episodes with Arbotix Robot Arm a distance of less than 5 cm was reached in 83% of the cases (4,198 episodes), and in 18% of the cases (916 episodes) a distance of less than 1 cm was reached. In the case of Dobot Mgcician Robot Arm simulation with 5,000 episodes, a distance of less than 5 cm was reached in 27% of the cases (1,374), but in no episode was a distance of less than 1 cm reached.

These results indicate that the trained model works well for the robot with which it was originally trained, but performs poorly when used with a different robot, even though both robots are 4-DOF, but differ in the size of the links as well as in the range in which the axes move, the q angles of each joint. If we want to use a model that is capable of controlling more than one robot we must use a larger data set containing the positions and angles of the different robots to be controlled.

In the future, we can implement our trained model in different robot arm making the adjustment in the DH parameters to compare the accuracy to reach the target position. The accuracy will be better with Robot Arm that has similar characteristic to ArbotiX Robot Arm. If we train the model with a bigger set of data we can have a good behavior and accuracy in the implementation. In this case to have a better performance for Dobot Magician Robot Arm, we need to collect the data through human demonstration of almost

every possible movement and record the end effector position of robot as well the angles of the joint.

ACKNOWLEDGMENT

Acknowledgments are due to the Intelligent Systems Lab at Sonoma State University for supporting this work.

REFERENCES

- [1] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," 2015. arXiv:1511.03791.
- [2] R. Jiang, Z. Wang, B. He and Z. Di, "Vision-Based Deep Reinforcement Learning For UR5 Robot Motion Control," 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 2021, pp. 246-250.
- [3] S. James and E. Johns, "3d simulation for robot arm control with deep q-learning," 2016. arXiv:1609.03759.
- [4] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2017, pp. 819–825.
- [5] D. K. et al., "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," CoRR, vol. abs/1806.10293, 2018. arXiv:1806.10293.
- [6] K. K. et al., "Leveraging deep reinforcement learning for reaching robotic tasks," in 2017 IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 490–491.
- [7] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," 2016. arXiv:1610.00633.
- [8] T. H. et al., "Composable deep reinforcement learning for robotic manipulation," in 2018 IEEE Int. Conf. on Robotics and Automation (ICRA), 2018, pp. 6244–6251.
- [9] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013. arXiv:1312.5602.
- [10] P. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," 2017. arXiv:1706.03741.
- [11] M. Vecerik et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2018. arXiv:1707.08817.
- [12] M. Flageat, K. Arulkumaran, and A. Bharath, "Incorporating human priors into deep reinforcement learning for robotic control," in ESANN 2020 proceedings, 2020.
- [13] T. Gao, "Optimizing robotic arm control using deep Q-learning and artificial neural networks through demonstration-based methodologies: A case study of dynamic and static conditions," Robotics and Autonomous Systems, vol. 181, 2024.
- [14] L. Graesser and W. Keng, Foundations of Deep Reinforcement Learning: Theory and Practice in Python. Pearson, 2019.
- [15] E. Akhtar, Practical Reinforcement Learning. Packt Publishing, 2017.
- [16] T. Beysolow, Applied Reinforcement Learning with Python. Apress, 2019.
- [17] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," 2016. arXiv:1511.06581.
- [18] I. Goodfellow et al., Deep Learning. MIT Press, 2016.
- [19] PhantomX reactor robot arm kit. [Online]. Available: <https://www.trossenrobotics.com/p/phantomx-ax-12-reactor-robot-arm.aspx>.
- [20] R. Schilling, Fundamentals of Robotics: Analysis and Control. Prentice-Hall, 1990.
- [21] M. Spong, S. Hutchinson, and M. Vidyasagar, Robot Modeling and Control. Wiley, 2020.
- [22] S. Gandhi, "Design, simulation and testing of a controller and software framework for automated construction by a robotic manipulator," Master's Thesis, Apr. 2019.
- [23] A. Gulli and A. Kapoor, TensorFlow 1.x Deep Learning Cookbook. Packt Publishing, 2017.
- [24] S. Fujimoto, D. Meger, and D. Precup, "An equivalence between loss 24functions and non-uniform sampling in experience replay," 2020. arXiv:2007.06049.
- [25] "Dobot magician user guide," 01/2019.
- [26] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," The International Journal of Robotics Research, vol. 40, no. 4, pp. 698–721, 2021.
- [27] S. Ravichandiran, Deep Reinforcement Learning with Python. Packt Publishing, 2020.