Deep Asynchronous Reinforcement Knowledge **(DARK)** *for* Embodied Intelligent Agent

Gyan Tatiya, Sambit Pradhan

1 Project Overview

Embodied Artificial Intelligent Agents are the next hyper-revolution in machine evolution, that may very well hold the key to the future of humanity. With unprecedented demand for advanced intelligent complex automation, mobile robotics, autonomous intelligent vehicles and applied artificial intelligence applications, Machine Learning and especially Reinforcement Learning is the new adventurous frontier in the AI gold rush. Why is Reinforcement Learning of such critical importance? Because it mimics and mirrors the natural learning process of all animate entities' by interacting and exploring their environments, receiving feedback, learning and accumulating knowledge over a period of time.

No where is the aforementioned aphorism more befitting than in the realm of Deep Reinforcement Learning (DRL) Neural Networks. DRLs' are target / goal-oriented algorithms, which self-learn to attain a complex objective (goal) or maximize particular dimension over several experience cycles.[1]: As our Final Project for Reinforcement Learning coursework we aspire to design, develop and implement a Deep Reinforcement Learning Neural Network which we call as DARK – Deep Asynchronous Reinforcement Knowledge – and augment DARK to simulate the training and behaviour of a realistic robotic arm model and require this embodied intelligent agent to solve realistic real-world approximate tasks. We have three precise aims for the project.

- **DARK:** Design and develop a Deep Reinforcement Knowledge Neural Network that can accomplish complex realistic real-world tasks such as object localization, object manipulation and object mobilization.
- **SIMULATION:** Design a simulation of high precision sparse reward DARK embodied agent for a high DOF robotic arm to accomplish the aforementioned real-world tasks.
- **KNOWLEDGE TRANSFER:** Time permitting, we want to cross train the embodied agent for different real-world tasks utilizing learned model knowledge transfer.

To achieve the above mentioned goals we implement a simulation of a high-DOF robotic arm to learn one or multiple of the following real-world tasks that include pushing, sliding and pick place with a Fetch robotic arm:

Target Object Localization (TOL): Move the robotic arm's end-effector to the desired goal position i.e. the object.
Target Object Shove (TOS): Move a object by pushing it until it reaches a desired goal position
Target Object Pick and Place (TOP): Pick up a object from the platform using its gripper and move it to a desired goal position.

All tasks have sparse binary rewards and follow a Deep Reinforcement Learning (DRL) framework. In all the tasks, the goal is 3-dimensional and describes the desired position of the object or the end-effector. Actions are 4-dimensional: 3 dimensions specify the desired gripper movement in Cartesian coordinates and the last dimension controls opening and closing of the gripper. We have selected the combination of four core components in order to implement the project:

Open AI Gym: Advanced Reinforcement Learning Environment.
Mujoco Simulator: Physics engine for simulation.
Reach Robotic Arm: Model of the Reach 7 DOF robotic arm.
TensorFlow: Deep learning framework.

2 Background and Related Work

Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning [1]. This is the seminal paper that presents the importance of complex synergies between non-prehensile (e.g. pushing) and prehensile (e.g. grasping) actions: pushing can help rearrange cluttered objects to make space for arms and fingers; likewise, grasping can help displace objects to make pushing movements more precise and collision-free. This paper was our primary source for inspiration.

Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research [2] The technically paper illustrates in detail the technical specifications of the Open AI environment and provides reference example of how to start with basic projects on which other advanced projects can be build upon.

3D Simulation for Robot Arm Control with Deep Q-Learning[3] The paper presents an approach which uses 3D simulations to train a 7-DOF robotic arm in a control task without any prior knowledge. The controller accepts images of the environment as its only input, and outputs motor actions for the task of locating and grasping a cube, over a range of initial configurations.

Robotic Arm Control and Task Training through Deep Reinforcement Learning[4] The paper presents an important detailed and extensive comparison of the Trust Region Policy Optimization and Deep Q-Network with Normalized Advantage Functions with respect to other state of the art algorithms, namely Deep Deterministic Policy Gradient and Vanilla Policy Gradient.

Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control [5] The Computer Vision (Machine) based paper proposes a method for deep reinforcement learning and develop a system for learning target reaching with a three-joint robot manipulator using external visual observation. A Deep Q Network (DQN) was demonstrated to perform target reaching after training in simulation.

Deep Reinforcement Learning for Robotic Manipulation-The state of the art[6] The primary focus of this work is to enumerate the various approaches and algorithms that center around application of reinforcement learning in robotic ma- nipulation tasks.

Evaluating Transfer Learning Methods for Robot Control Policies[7] This paper has two important components it compares and contrasts reinforcement and supervised learning systems and evaluates three transfer learning techniques in a simple two dimensional environment. The project finds that within the 2D environment a simple reinforcement model performs a better than a more complex supervised one and that Progressive networks are the most powerful of the three transfer learning methods.

3 Problem Formulation and Technical Approach

The agent interacts with a simulated environment E in a sequence of actions, observations and rewards. At any given time step, the agent chooses an action at to perform in the environment from the set of valid actions, $\mathcal{A} = \{a_1,\ a_2\ a_3\ ...\ ,\ a_K\}$. The action is executed in the simulated environment E and the internal state of the environment changes and a reward is returned. The agent receives a set of observations and a reward rt form the simulated environment E.

The main goal of the agent is to interact with the simulated environment E by choosing actions in a way that maximizes future rewards. The return is denoted by

$$R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r_i$$

where T is the time-step at which the interaction terminates and γ is a discount factor for future rewards. The optimal action-value function $Q^*(s, a)$ is defined to maximum expected return achieved by following any policy after experiencing a sequence s and executing some action a,

$$Q^*(s,\ a) = max_{\pi} \mathrm{E}[R_t | s_t = s, a_t = a, \pi]$$

where π is a policy mapping sequences to actions.

The optimal action-value function maximizes the expected value $r+\gamma Q^*(s', a')$, using the optimal value $Q^*(s', a')$ of the sequence s' at the next time-step for all possible actions a'.

$$Q^*(s,\ a) = \mathrm{E}_{s' \sim \mathcal{E}}\ [r + \gamma \max_{a} Q^*(s',\ a') | s,\ a]$$

To estimate the action-value function, we will use non-linear function approximator, deep Q-network with weights θ. Deep Q-network is trained by optimizing the loss function $L_i(\theta_i)$ that changes at each iteration i. Some commonly use loss functions are root mean square error and cross entropy. We will conduct experiments to figure out the best possible loss functions that optimizes the deep Q-network. We will optimize the loss functions by gradient descent as mentioned in [8].

4 Evaluation and Expected Outcomes

We will use two metrics to evaluate the performance of the agent. The first metric is the total reward collected by the agent in an episode averaged over a number of episodes computed during training. We expect to see consistent increase in the reward while training period. The second metric is the estimated Q value by the policy's action-value function, which informs that how much discounted reward the agent can achieve by following its policy for a given state during training period. Similar to the first metric, we expect the Q value to grow consistently while training period.

References

[1] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv preprint arXiv:1803.09956*, 2018.

[2] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[3] Stephen James and Edward Johns. 3d simulation for robot arm control with deep q-learning. *CoRR*, abs/1609.03759, 2016.

[4] Andrea Franceschetti, Elisa Tosello, Nicola Castaman, and Stefano Ghidoni. Robotic arm control and task training through deep reinforcement learning.

[5] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[6] Smruti Amarjyoti. Deep reinforcement learning for robotic manipulation-the state of the art. *arXiv preprint arXiv:1701.08878*, 2017.

[7] Rad Ploshtakov, Edward Johns, and Stefan Leutenegger. Evaluating transfer learning methods for robot control policies. 2017.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.