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

Gyan Tatiya, Sambit Pradhan

1 Abstract

Advanced Embodied Artificial Intelligent Agents with high degree of freedom and complex articulation such as anthropomorphic humanoid robots present an exciting training opportunity using Deep Reinforcement Learning paradigm. Deep Reinforcement Learning methods facilitate creation of complex action policy for such advanced physiology structures that may result in superior learning, reward generation and post learning performance. However, training of such complex policy can take a long time usually amounting to years of experience. We propose transferring knowledge from a learned agent to an untrained agent to improve the performance in early training time.

2 Introduction

In the current project, Deep Asynchronous Reinforcement Knowledge (DARK) for Embodied Intelligent Agent, we train two Humanoid agents to perform two specific locomotion tasks a) Humanoid-v2 - Walking from initial stationary standing position, b) HumanoidStandup-v2 - Standing upright from initial laying on the ground position. Furthermore, the agent was transformed into a a) Underweight and b) Overweight agent. We train the two morphed agents, that are derivatives of the standard Humanoid-V2, but with altered morphology, and observe their behaviour (shown in Figure 1). Additionally, we perform transfer learning between the two agents with domain adaptation where learned model of one agent is exploited to improve generalization in another setting and performance in the other agents. The core reinforcement learning algorithm used is Proximal Policy Optimization (PPO), which is a policy gradient method that learns online while the agent is interacting with the environment. To achieve the objective of this project, we performed the following tasks:

- **DARK:** Design and develop a Deep Reinforcement Knowledge Neural Network that can train complex highly articulated simulated Humanoid

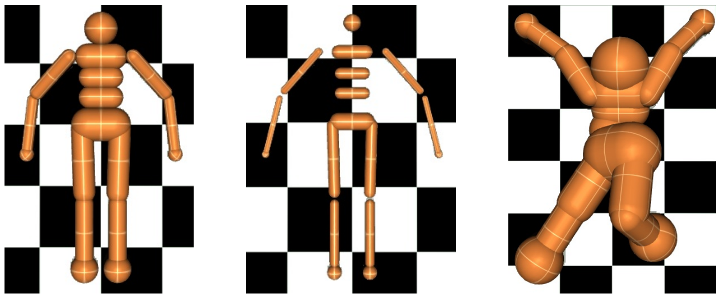
Figure 1: Humanoid (Standard), Humanoid (Underweight), and Humanoid (Overweight)

agents locomotion related tasks, namely, standing upright and walking.

- **TRANSFER LEARNING:** We augment the learning of one agent by transferring the knowledge gained by one agent to the another. We perform transfer learning to improve learning in a new task through the transfer of knowledge from a related task that has already been learned.
- **SCALE INVARIANT FEATURE SPACE REINFORCEMENT LEARNING** We train morphologically variant agents with the same number of articulation and physiology but of different dimensions - the same task and transfer knowledge in-between the agents. Thus performing Scale Invariant Feature Space Reinforcement Learning.

3 Background and Related Work

In robotics and reinforcement learning, prior works have considered building direct isomorphisms between state spaces. However, most of these methods require specific domain knowledge to determine how to form the mapping, or operate on simple, low-dimensional environments. Some aspects of the skill may not be transferable at all, in which case they must be learned from scratch, but we would like to maximize the information transferred between the agents. Some of the research and publication that we followed are mentioned herein:

Bipedal Walking Robot using Deep Deterministic Policy Gradient [1]. In this paper, the authors present an architecture to design and simulate a planar bipedal walking robot (BWR) using a realistic robotics simulator, Gazebo. The robot demonstrates successful walking behaviour by learning through several of its trial and errors, without any prior knowledge of itself or the world dynamics. The autonomous walking of the BWR is achieved using reinforcement learning algorithm called Deep Deterministic Policy Gradient (DDPG). DDPG is one of the algorithms for learning controls in continuous action spaces.

Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning [2]. In this paper, the authors formulate this multi-agent transfer learning problem in a setting where two agents are learning multiple skills. Using the skills that have been already acquired by both agents, each agent can construct a mapping from their states into an invariant feature space. Each agent can then transfer a new skill from the other agent by projecting the executions of that skill into the invariant space, and tracking the corresponding features through its own actions.

Evaluating Transfer Learning Methods for Robot Control Policies [3]. This paper has two important components that 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.

4 Problem Formulation and Technical Approach

4.1 Environment

We used OpenAI Gym [4], which is advanced Reinforcement Learning Environment. OpenAI Gym requires MuJoCo [5] in the backend, which is physics engine for simulation. For implementing neural network we used TensorFlow [6], which is a deep learning framework.

4.2 Agent

We used two environments namely 'Humanoid-v2' and 'HumanoidStandup-v2'. Humanoid-v2 is a three dimensional bipedal agent. The goal here is to make the agent walk forward. HumanoidStandup-v2 is also a three dimensional bipedal agent that is supposed to stand-up. For both the agents, the action space is described by a vector of 17 real numbers representing abdomen, hips, right and left arms, legs, knees etc. and observation space is described by a vector of 376 real numbers denoting position, orientation, rotation, velocity, force etc. The of the agents is computed by:

$$\begin{aligned} Reward \; = \; & Linear\ Velocity\ Costs \; - \; Quad\ Control\ Cost \; - \\ & Quad\ Impact\ Cost \; + \; Alive\ Bonus \end{aligned}$$

More details about the reward can be found on GitHub repository of OpenAI Gym [4].

4.3 Learning Methodology

For training the agents, we used PPO, discussed below, and for knowledge transfer to an agent, we initialized the network weights of a trained agent before its training starts.

4.3.1 Proximal Policy Optimization (PPO)

For training the agents we used Proximal Policy Optimization (PPO) [7], a reinforcement learning algorithm designed in OpenAI in 2017. PPO is the default reinforcement learning algorithm at OpenAI because it performs comparable or better than state-of-the-art approaches in on continuous robotic control tasks.

4.3.2 General Policy Gradient Loss

A general Policy Optimization method start by defining the policy gradient loss as:

$$L^{PG}(\theta) = \hat{\mathrm{E}}_t[\log \pi_\theta(a_t|s_t)A_t]$$

It is expectation over the log of policy actions times estimate of advantage function. π_θ is the policy. It's a neutral network that takes the observed states from the environment as input and suggests actions to take as an output. Advantage function A_t, estimates what is the relative values of the selected actions is in the current state. A_t is computed using two terms: Discounted sum of rewards and baseline estimate/value function:

$$A_t \;=\; Discounted\, rewards \;-\; Baseline\ estimate$$

Discounted rewards is the weighted sum of all the rewards the agent got in the current episode. Discounted rewards are calculated after the episode sequence was collected from the environment. So, all the rewards are known and there is no guessing involve in computing the discounted return. Baseline estimate gives an estimate of the discounted sum of the rewards. Basically, it is trying to guess what the final return is going to be in this episode. It is computed using neural net, so it will be an approximate value. If the advantage action is positive, that means the actions that the agent took resulted in better than average return. So, the probability of those actions is increased for the future when the same state is encountered. Similarly, if the advantage action is negative, the probability of those actions is reduced.

4.3.3 Trust Region Policy Optimization Loss

PPO is a based on another algorithm called TRPO [8]. This is the TRPO objective function:

$$L^{CPI}(\theta) = \hat{\mathrm{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}A_t] = \hat{\mathrm{E}}_t[r_t(\theta)A_t]$$

It is similar to general policy gradient loss, but instead of log of policy actions, it has probability ratio ($r_t(\theta)$). Probability ratio is the ratio between the updated policy outputs and the outputs of the previous old policy. $r_t(\theta)$ values will be larger than 1, if the action is more like now than the old version of policy. $r_t(\theta)$ values will be between 0 and 1, if the action is less likely now than old policy.

TRPO adds a KL constraint to optimization objective. KL make sure that new updated policy does not move too far away from the old policy. This may lead to undesirable training behavior. PPO adds this extra constraints directly into this objective function.

4.3.4 Proximal Policy Optimization Loss

The the main objective function of PPO is defined as:

$$L^{CLIP}(\theta) = \hat{\mathrm{E}}_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)]$$

It is the expectation over the minimum of two terms. The first term is $r_t(\theta)$ times the advantage estimate. This is the default objective for normal policy gradient which pushes the policy towards actions that get a high positive advantage over the baseline. The second term is similar to the first term except that it contains a truncated version of $r_t(\theta)$ ratio by applying a clipping operation between 1 - ϵ, 1 + ϵ. Where ϵ is 0.2. Clipping operation removes the incentive for moving $r_t(\theta)$ outside of the interval [1 - ϵ, 1 + ϵ].

In the PPO algorithm, the current policy interacts with the environment generating episode sequence for which the advantage function is calculated using the baseline estimate. Then this experience is used to run gradient descent on policy network using the clipped PPO objective. For the experiments, we have adapted the PPO implementation from Stable Baselines [9], which is a fork of OpenAI Baselines.

5 Experiments and Results

We are measuring three criterias: We document the learning curve of each agent that graphically shows the reward obtained (Y-axis) for each time step (X-axis), we compute the number of actions the agents takes in an episode for Humanoid-v2, and we observe the physical behaviour in the simulation environment.

5.1 Experiment 1: Conception Reinforcement Training of Humanoid-V2 Agents

In this experiment the two agents were taught their respective tasks from the conception (beginning). a) Humanoid-v2 - Walking from initial stationary standing position. b) HumanoidStandup-v2 - Standing upright from initial laying on the ground position.

5.1.1 Results of Experiment 1

a) The learning curve for Humanoid-v2 is shown in Figure 3 at Top left (a) with blue color. The curve starts to converge after 2 million time steps. After training, the agent was able to take maximum of 191 actions before episode ends. An untrained agent was able to take only 50 steps. In addition, we observed

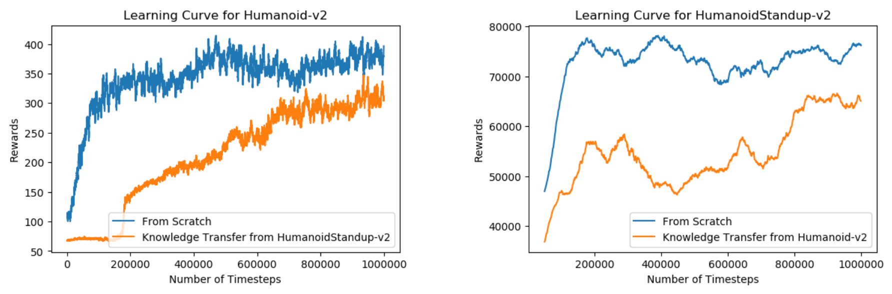

Figure 2: Knowledge Transfer from HumanoidStandup-v2 to Humanoid-v2 (left) and Knowledge Transfer from Humanoid-v2 to HumanoidStandup-v2 (right)

that as compared to the untrained agent, the trained agent learned to move its hands to keep the body balanced while walking.

b) The learning curve for HumanoidStandup-v2 is shown in Figure 4 at Top left (d) with blue color. The curve keeps oscillating through out 10 million time steps. The trained agent learned to keep the hands in a specific position while trying to getup, whereas as an untrained agent keeps shaking both hands as well as legs.

5.2 Experiment 2: Inter Task Knowledge Transfer

In this experiments, we first trained the agents to achieve the specified task of their respective environments. Then we transferred their knowledge to the agent with different task. More specifically, we trained Humanoid-v2 and HumanoidStandup-v2 from scratch, and transferred HumanoidStandup-v2's knowledge to Humanoid-v2, and Humanoid-v2's knowledge to HumanoidStandup-v2.

5.2.1 Results of Experiment 2

The learning curve for both the agents is shown in Figure 2. For both the tasks, knowledge transfer resulted in poorer performance as compared to the agents trained from scratch in terms of rewards over training time steps. Basically, it means that learning how to walk does not significantly help in learning how to stand-up, and similarly, learning how to stand-up does not help in learning how to walk.

5.3 Experiment 3: Scale Invariant Knowledge Transfer

In this experiments, we first trained the agents of a specific morphology and then transferred its knowledge to other agents with altered morphology within same task domain. a) We trained Humanoid-v2 (Standard) from scratch and transferred its knowledge to Humanoid-v2 (Underweight). b) We trained Humanoid-v2 (Standard) from scratch and transferred its knowledge to Humanoid-v2

(Overweight). c) We trained Humanoid-v2 (Underweight) from scratch and transferred its knowledge to Humanoid-v2 (Standard). d) We trained HumanoidStandup-v2 (Standard) from scratch and transferred its knowledge to HumanoidStandup-v2 (Underweight). e) We trained HumanoidStandup-v2 (Standard) from scratch and transferred its knowledge to HumanoidStandup-v2 (Overweight). f) We trained HumanoidStandup-v2 (Underweight) from scratch and transferred its knowledge to HumanoidStandup-v2 (Standard).

We did not transferred knowledge of Humanoid-v2 (Overweight) and HumanoidStandup-v2 (Overweight) to any other agents because of its poor performance.

5.3.1 Results of Experiment 3

The learning curve for (a), (b) and (c) experiments are shown in Figure 3. For all the curves, the rewards converge after 2 million time steps. The trained agent for (a) was able to take maximum of 148 actions before episode ends, (b)'s agent takes 18 actions and (c)'s agent takes 175 actions steps. Their respective untrained agents was able to take 64, 49 and 50 actions. Thus, the except (b), the agents learns to survive in the environment for longer than untrained agents. In addition, agents that uses knowledge of an agent with different morphology was able to get more rewards initially during training, except (b). This shows that knowledge transfer provides by agent of different morphology with better acting policy. The reason for (b) not performing well could be due to agents heavy weight, it was not able to make its body walk and fall immediately.

When trained from scratch, Humanoid-v2 (Underweight) shakes its body more than Humanoid-v2 (Standard) probably because of its lighter weight its actions tends to move its body more. In experiment (a), when Humanoid-v2 (Underweight) learns from Humanoid-v2 (Standard), we observed that the agent was walking much more stably. Similarly, in experiment (c), when Humanoid-v2 (Standard) learns from Humanoid-v2 (Underweight), we observed that the agent was shaking more while walking. This shows that knowledge transfer affects action policy of the target agent based on the source agent.

The learning curve for (d), (e) and (f) experiments are shown in Figure 4. The curves does not converge even after 10 million time steps. Agents that uses knowledge of an agent with different morphology was able to get more rewards initially during training. However, except (e), the performance is poor after knowledge transfer as compared to the agent trained from scratch. One possible reason could be that because we initially used an optimal policy of an agent with another morphology, it might get stuck in a local minima and not able to come out of it during training. These results shows that knowledge transfer may not necessarily be beneficial for every tasks.

In term of observation, we observe same behavior we observed for (a) and (c) in (d) and (f), respectively. In (d), when HumanoidStandup-v2 (Underweight) learns from HumanoidStandup-v2 (Standard), we observed that the agent was is much more stable. Similarly, in experiment (f), when HumanoidStandup-v2 (Standard) learns from HumanoidStandup-v2 (Underweight), we observed that the agent was moving its body more.

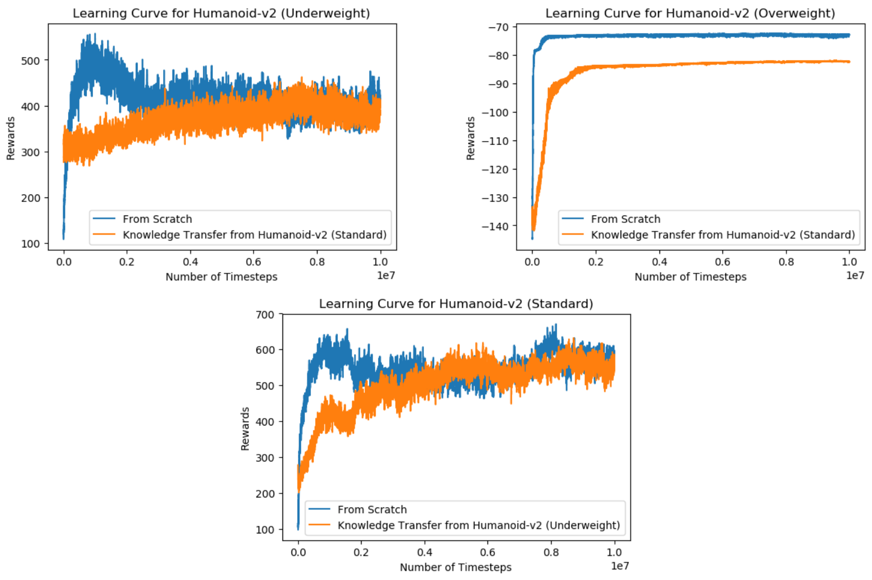

Figure 3: Top left (a): Knowledge Transfer from Humanoid-v2 (Standard) to Humanoid-v2 (Underweight), Top right (b): Knowledge Transfer from Humanoid-v2 (Standard) to Humanoid-v2 (Overweight), Bottom (c): Knowledge Transfer from Humanoid-v2 (Underweight) to Humanoid-v2 (Standard).

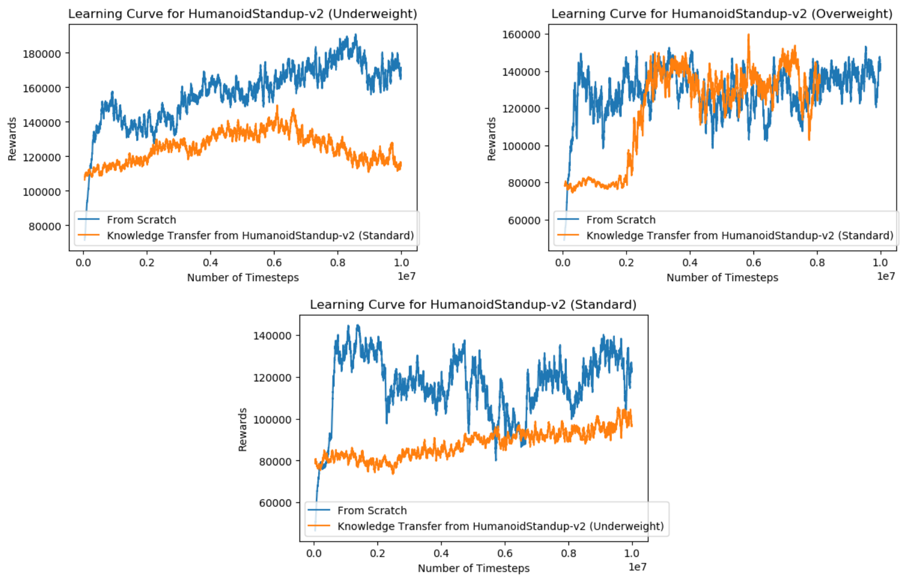

Figure 4: Top left (d): Knowledge Transfer from HumanoidStandup-v2 (Standard) to HumanoidStandup-v2 (Underweight), Top right (e): Knowledge Transfer from HumanoidStandup-v2 (Standard) to HumanoidStandup-v2 (Overweight), Bottom (f): Knowledge Transfer from HumanoidStandup-v2 (Underweight) to HumanoidStandup-v2 (Standard).

6 Conclusion and Future Work

In this work, we trained two agents, Humanoid-v2 and HumanoidStandup-v2, to solve a continuous control task. Results shows that the trained agent learns to get higher rewards and survive for longer than untrained agent. We showed that transferring knowledge across agents of different morphologies not only improve the performance initially during the training, but also adapt action policy of the source agent in the target agent.

In future, we are interested in trying out different deep reinforcement learning algorithms to solve achieve the goal of the environment. Because MuJoCo requires license, we were not able to experiment with more training methods and hyper-parameters as we activated a student license on a single computer. In addition, we would also like to chance the environment around the agent and perform transfer knowledge. For example, we can add rough terrain and train the agent to walk by using the knowledge on an agent who can walk in a plain terrain.

References

[1] Arun Kumar, Navneet Paul, and S. N. Omkar. Bipedal walking robot using deep deterministic policy gradient. *CoRR*, abs/1807.05924, 2018.

[2] Abhishek Gupta, Coline Devin, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *CoRR*, abs/1703.02949, 2017.

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

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[5] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[6] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[9] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.