Title: Transfer Learning between Control Tasks using Reinforcement Learning

G009 (s1876926, s0908445, s1873615)

Abstract

In this paper we test the state-of-the-art reinforcement learning algorithm Proximal Policy Optimisation (PPO) in the robotic control domain for their ability to transfer between similar, albeit different, tasks. We will use OpenAI's Bipedal walker's two environments; with the aim to reduce training times and improve performance comparative to training from scratch for both tasks. Our experiments show that weight sharing with all layers transferred can increase the initial level of performance - when transferring to both more complex or simpler tasks. We also show that training on multiple tasks can significantly increase performance on complex environments. These results show us that similar methods could be used to prepare an agent for very complex task by training on a simpler task. Thus, speeding up learning of a complex task and increasing the performance level.

1. Introduction

Reinforcement learning (RL) involves training an agent through punishment and reward without the need to specify exactly how the agent should approach the task Kaelbling et al. (1996). RL has achieved a high success in various challenges with a vast amount of research going into different algorithms and training techniques.

One field RL has been very successful in is in learning complex behaviours in robotic control. Robotic control tasks are notably difficult because of the controllers requirements to perform in high dimensional state spaces, deal with uncertainties and function on unseen instances (Kalashnikov et al., 2018). Tasks like walking or manipulating objects in ones hand have a large number of applications and are therefore important tasks to solve.

However, experimenting with robots in the real world is also an expensive and time consuming task which has led many researchers to train and test on simulated robotic environments. The amount of research on reinforcement learning and simulated robitics has led a number of companies to provide pre-built environments for running such experiments. Mujoco[1] and OpenAI both provide test environments for simulated robots, from 2D walkers to 3D hands. Algorithms such as DDPG, DQN and HER have all been applied to the above problems[2] but recently Proximal Policy Optimisation (PPO) has emerged as the state of the art on tasks such as a 2D walker Zhang & Zaiane (2017). However, this algorithm still struggles to achieve a high reward on more complex tasks and we therefore aim to explore techniques to bridge the performance gap between simple and more complex tasks.

Transfer Learning (TL) offers researchers a paradigm to generalise to new scenarios with minimal training whilst enhancing performance. It will be an important research direction if we're going to see machines exhibit general intelligence and be able to learn a myriad of different tasks previously not countered. TL has previously been used to transfer policies trained in a simulated environment into a real robot, bridging the simulation-reality gap. TL has also been used to transfer policies for the same task between a 3 degrees-of-freedom argm and a 4 degrees-of-freedom arm (Kojcev et al., 2018). Outside the study of robotics, transfer learning has been used to transfer knowledge from source tasks to target tasks using Atari games successfully, they used a single DQN source network trained upon multiple games which could then transfer to other Atari tasks (Parisotto et al., 2015).

However, despite the promising results from new RL algorithms on continuous control tasks, TL is yet to have been used to transfer RL policies between different tasks. The ability for a simulated robot to transfer between different tasks will make real robots attempting new tasks safer and faster to reach a level of performance more effective for real world usage.

Therefore, in this paper we aim to find a form of transfer learning that improves the performance of an RL agent on the given tasks. To do this we will use the Proximal Policy Optimisation (PPO) algorithm Schulman et al. (2017) on OpenAI's Bipedal walker and Bipedal Walker Hardcore environments. We will explore methods of transfer learning such as weight sharing Tirinzoni et al. (2018) and multi-task learning Parisotto et al. (2015), to asses their abilities to improve an agent's performance.

Our experiments found that weight sharing could increase the initial reward but would struggle to significantly increase the maximum reward achieved by the more complex environments. However, multi-task learning - training an agent on alternating different tasks - achieved a significant improvement in the more complex tasks. We believe this could be because in more complex tasks, they're less

[1] http://www.mujoco.org

[2] http://gym.openai.com

predictable and the multi-task training helps the agent generalise better.

In the next section we will discuss the task, algorithm and environments in detail. From there we will discuss our methodology, including definitions of reinforcement and transfer learning. We will then discuss our experiments setup and results followed by a brief mention of related work. We will then conclude with a summary of our results and findings followed by a mention of potential future areas of research.

It should be noted that our report differs considerably to our previous report, Coursework 3, where we outlined an approach for transferring between OpenAI's robot dexterity tasks with a focus on reducing long training times found in using RL in robot control. This is because of technical difficulties installing Mujoco on the MLP Cluster and the time constraints at hand. We considered other robotics environments such as RoboSchool but without success, so we have focused on the Box2d environments which are the closest to a robots control problem with its rigid body physics and dynamics (Catto, 2013). Our previous proposal can stand as a direction of future work.

2. Task and data

In order to research transfer learning between environments, we needed to use continuous control environments with similar action spaces and inputs. These environments must also be comparable to more complex robotic simulation tasks, such as the Mujoco dexterity environments, in order for the results to be applicable to a wider set of problems. We therefore used the *BipedalWalker-v2* and *BipedalWalkerHardcore-v2* environments of OpenAI Gym [3] which are based on Box2D physics engine[4].

The first environment is the *BipedalWalker-v2* where the agent needs to travel across small random variations of a flat terrain as in figure 2(a). The second one is more challenging which is *BipedalWalkerHardcore-v2*. The agent is required to navigate across an environment of randomly generated terrain within a time limit, without falling over (figure 1(b)). The BipedalWalker character has 4 degrees of freedom, 2 hip and knee joints as shown in figure1(b). In both the environments, the state consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements. There are no coordinates in the state vector.

Reward is given for moving forward to a total of 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small amount of points and so the more optimal the agent, the higher the score. (Ha (2018))

These environments provide a continuous control task on a simulated robot, within a simulated physics environment. These environments are therefore comparable to more complex robotic simulation environments but provide a less computationally intensive environment in which to perform our research.

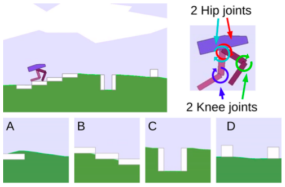

Figure 1. BipedalWalkerHardcore

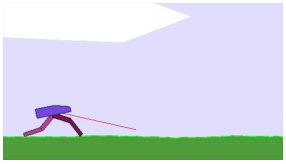

Figure 2. BipedalWalker

OpenAI Gym also provides a toolkit for developing and comparing reinforcement learning algorithms scientifically (Brockman et al. (2016)), including PPO, DDPG, DQN and TRPO[5].

Using the above environments, our task is to simulate an agent using PPO2 (Schulman et al. (2017)) before comparing the results to the performance of models that have been pre-trained. Pretraining our networks will involve a number of different weight sharing techniques, as well as a form of multi-task learning where we will train our agent on multiple tasks simultaneously.

We will measure the success of our model by using the episode reward mean (ERM). This is the mean reward the agent achieved over each episode in an update. To evaluate our implementations we will look at the zero shot performance (or initial skill level) after transferring the weights to the new task environment and the rate of improvement of the reward, which we hope will be higher than our benchmarks. Finally, we will look at the maximum mean reward per episode of our models to evaluate the level of their best performance

3. Methodology

3.1. Reinforcement Learning

Our environment will be a Markov Decision Process represented by a tuple $M = (\mathcal{S}, \mathcal{A}, ,\mathrm{T}, \mathcal{R}, \mathcal{S}_0, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is the action space, $p(s_{t+1}|s_t, a_t)$ is the transition probabilities over the transition space T, $\mathcal{R}$ is a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, $\mathcal{S}_0$ is the initial state distribution, and $\gamma \in [0, 1)$ the discount factor.

A deterministic policy is a mapping from states to actions $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Every episode starts with sampling an initial state s_0. At every time step t the agent produces an action based on the current state $a_t = \pi(s_t)$. Then it gets the reward $r_t = r(s_t, a_t)$ and the environment's new state is sampled from the distribution $p(\cdot|s_t, a_t)$. A discounted sum of future rewards is called the *return* $\mathcal{R}_t = \sum_{i=t}^{+\infty} \gamma^{t-i} r_i$. The agent's goal is to maximise its expected return $\mathbb{E}_{s_0}[\mathcal{R}_0|s_0]$. The Q-function or action-value function under a policy π is defined

[3] http://gym.openai.com
[4] https://gym.openai.com/envs/#box2d
[5] https://github.com/openai/baselines

as $Q_\pi(s_t, a_t) = \mathbb{E}[\mathcal{R}_t|s_t, a_t]$.

Let π^* denote an *optimal policy* i.e. any policy π^* s.t. $Q_{\pi^*}(s, a) \geq Q_\pi(s, a)$ for every $s \in \mathcal{S}, a \in \mathcal{A}$ and any policy π. All optimal policies have the same Q-function which is called the *optimal Q-function* and denoted Q^*. It is easy to show that it satisfies the *Bellman equation*:

$$Q^*(s, a) = \mathbb{E}_{s' \sim p(\cdot|s,a)}[r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')] \quad (1)$$

3.2. PPO2

The Proximal Policy Optimization algorithm performs comparably or better than state-of-the-art approaches while being much simpler to implement and tune. PPO has become the default reinforcement learning algorithm at OpenAI because of it's ease of use and good performance Dhariwal & Wu (2017). PPO adds a soft constraint that can be optimized by a first-order optimizer. The agent may make some bad decisions once in a while but it strikes a good balance on the speed of the optimization. Experimental results prove that this kind of balance achieves the best performance with the most simplicity Schulman et al. (2017).

The Proximal Policy Optimization algorithm combines ideas from A2C (having multiple workers) and TRPO (it uses a trust region to improve the actor). The main idea is that after an update, the new policy should be not too far form the old policy. For that, PPO uses clipping to avoid too large update Hill et al. (2018). The PPO2 formula as purposed by Schulman et al. (2017) is the following:

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\,\hat{A}_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t] \quad (2)$$

Where epsilon is a hyper-parameter, say, $\epsilon = 0.2$ The motivation for this objective is as follows. The first term inside the min is L^{CLIP}. The second term, $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval $[1-\theta, 1+\theta]$ Finally, we take the minimum of the clipped and unclipped objective, so the final objective is a lower bound on the unclipped objective. Consequently, we have two cases to consider as shown in figure 3.

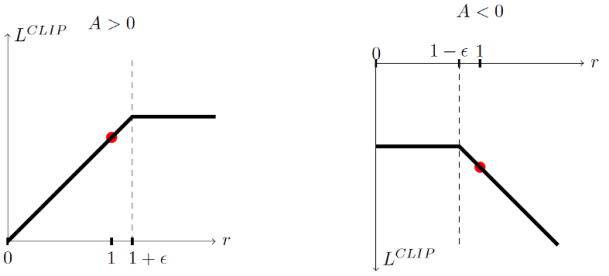

Figure 3. Plots showing one term of the surrogate function L^{CLIP} as a function of the probability ratio r_t, for positive advantages(Case 1) on left and negative advantages(case 2) on the right. The red circle on each plot shows the starting point for the optimization,i.e..., $r = 1$. Note that L^{CLIP} sums many of these terms.

The first case is when the advantage is > 0. If $\hat{A}_t > 0$, it means that the action is better than the average of all the actions in that state. Therefore, we should encourage our new policy to increase the probability of taking that action at that state. Consequently, it means increasing r_t, because we increase the probability at new policy (because A_t * new policy) and the denominator old policy stays constant. The second case is when the advantage is < 0. If $\hat{A}_t < 0$, the action should be discouraged because negative effect of the outcome. Consequently, r_t will be decreased (because action is less probable for current policy than for the old one) but because of the clip, r_t will only decreases to as little as $1 - \theta$ (Schulman et al. (2017)).

3.3. Transfer learning

The idea behind transfer learning is you can use a previously trained network to transfer to a similar, but different, task - improving the generalisation in the new setting. Results have shown that they can be effective at speeding up learning when transferring to related but distinct RL tasks (Konidaris & Barto, 2006). Which is especially a prevalent concern of the robot learning community where training times of reinforcement learning have taken several months (Kalashnikov et al., 2018). We can formally define transfer learning in Definition 3.1.

Definition 3.1 (Transfer Learning) *Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$ and $T_S \neq T_D$.*

In our case, our source domain will be one of *BipedalWalker-v2* and *BipedalWalkerHardcore-v2* and we will be transferring to the other. From the simpler task to the more complex task and vice versa.

If we were to train two seperate networks for two similar tasks there would be a *sharing* between the weights of the lower levels. Rather than train a full network independently, we can take the weights from the first task, transfer them to a new network and train just the last layers to generalise to the new task. This approach to transfer learning is called *weight sharing*, and we will be investigating its utility in our experiments.

Another method is to train on multiple different environments and to then to transfer to another environment (Ciosek & Whiteson, 2017). Which has been shown experimentally to learn better and faster than a policy gradient baseline. We will be alternating between *Bipedal Walker* and *Bipedal Walker Hardcore* and then testing on both environments.

From a successful implementation of transfer learning you can expect to see:

- The initial skill (before refining the second network) on the second task is higher than it otherwise would

be.

- The rate of improvement of skill during training of the second task is higher than it otherwise would be.
- The skill level of the secondary trained model after converging is greater than it otherwise would be.

4. Experiments

4.1. Baseline experiments

The aim of our experiments were to investigate the potential benefits of transfer learning between tasks. Does transferring speed up the learning in the second task? How does it affect the initial skill? We also want to investigate the effects of transferring to a simpler task (Bipedal → Bipedal Hardcore) and vise versa.

To analyse the effects we started out investigating how tasks independently trained to find a baseline for both the Bipedal and the Bipedal Hardcore environment. Providing a comparison to test the utility of our hypotheses.

To do this we used the PPO algorithm discussed above on both the Bipedal Walker and the Bipedal Walker Hardcore using the OpenAI baseline PPO2 model [6] using the 'MLP' architecture consisting of 2 hidden layers of 64 units and a Tanh activation unit.

Simulations are inherently deterministic with their seed numbers being the only element of randomness and uncertainty. If the seed number is unchanged, with enough training time, the model will memorise the random patterns making the learn policy brittle as it over fits (Zhang et al., 2018). To ensure our models didn't exhibit over fitting, which could possibly diminish performance once transferred to a new task, we aimed to sample the seed numbers from a sample size of 10. Unfortunately, after having technical difficulties with our installation on the MLP Cluster and having to change our project there wasn't enough time to do so. Nevertheless, we would like to do this in future work averaging over a number of seed numbers would increase the reliability of our findings and prevent over fitting to subtleties in an inherently deterministic environment.

Due to the large number of hyper-parameters and the complexity in both tasks a thorough hyper-parameter grid search is beyond realistic under the time and computing resource constraints. For both the value function and policy function we used the Adam optimiser (Kingma & Ba, 2014). The following paper was our guide for hyper-parameter setting as they used PPO with the Walker2D environment very similar to Bipedal walker and achieved state of the art results (Schulman et al., 2017). In our experiments we used the hyper-parameters in Table 1. Our weights were initialised to 0.003 multiplied by a normalized uniform distribution.

Our results show that both walkers begin with mean reward of roughly -100 however, the Bipedal Walker converges to a maximal episodic mean reward score of 300. See Figure 4. The Bipedal Walker Hardcore seen in Figure 4 cannot achieve a score greater than -40 after its training. Our hope is that transferring the weights will increase this score in a shorter training time.

[6] https://github.com/openai/baselines/ppo2

Hyper-parameter	Value
Learning rate	0.001
Discounting Factor	0.99
Batches per Update	64
Clipping (ϵ)	0.2
Total Timesteps	10,000,000
Steps per Update	2048
Epochs per Update	10
Discount (γ)	0.99
GAE parameter (λ)	0.95

Table 1. Hyper-parameter settings.

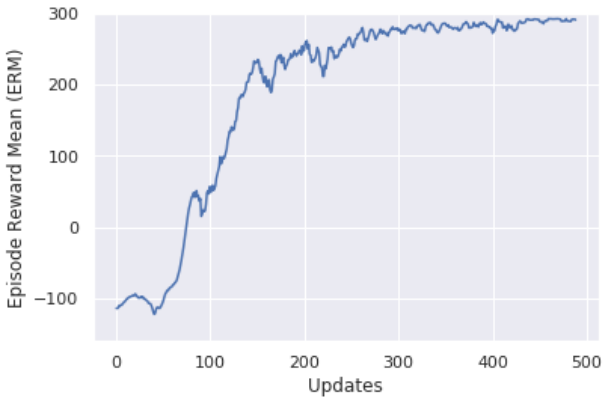

Figure 4. Baseline for normal Bipedal Walker. Showing episodic mean reward by n updates.

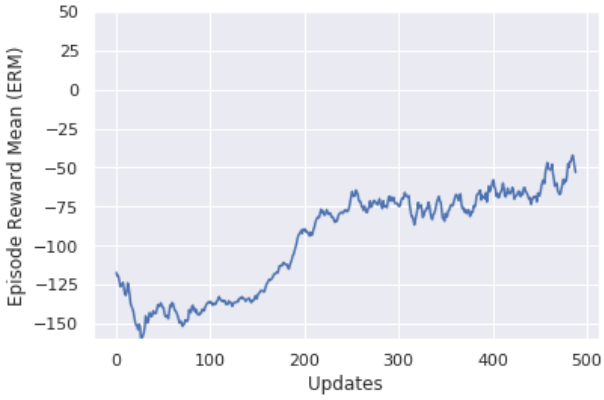

Figure 5. Baseline for hardcore Bipedal Walker. Showing episodic mean reward by n updates.

Our interpretation from the baselines is that there is a large scope for the Bipedal Hardcore environment to see benefits from our transfer learning approaches. Especially in the first 150 updates where the performance dips below the randomly initialised weights performance. As well as the

overall hardcore performance since doesn't exceed a reward of -40. This suggests that the hardcore environment is too complex a task for the agent to learn from scratch.

4.2. Weight Sharing

The aim of these experiments was to see if transferring the weights to the more complex tasks would help performance and to see if simpler environments could be made more robust using the weights transferred from a more complex environment.

All the hypermeters were the same as the baseline experiment run for a total timesteps of 10 million for all the weight sharing experiments.

We verified our implementation of transferring weights by transferring previous learnt weights in *Bipedal Walker* back to *Bipedal Walker* to check the ERM continued to have a performance near to 300. Our experiment verified our transfer learning worked albeit the ERM didn't converge as in the *Bipedal Walker* baseline. We suspect this might to do with the Adam Optimiser hyper-parameters, after the transfer, the learning rate would have been increased back to a large value making the policy jump to a less optimal value.

4.2.1. Weight sharing: 1st layer

First, We wanted to know what conditions transfer learning via weight sharing is most effective. This meant experimenting with the number of layers we transfer, and what weights we train thereafter. We tried transferring the weights of only the first layer between the environments. In the first experiment, we started by sharing the first layer weights of *BipedalWalkerHardcore* to the *BipedalWalker* environment. The second experiment, we transferred the first layer weight of *BipedalWalker* to the *BipedalWalkerHardcore* environment.

The findings of the both experiments can be seen in figure 6 for the *BipedalWalker* and figure 7 for the *BipedalWalkerHardcore*. These results show us that the performance was almost the same as the performance of the baseline with a slight improvement. For the *BipedalWalker*, the first 110 updates achieved an ERM that was slightly higher than the baseline, with a difference of almost 60 ERM. While for the *BipedalWalkerHardcore*, a marginally higher ERM was seen in the first 200 updates for the environment in comparison to the baseline.

These results confirm that only transferring the first layer does not transfer enough information to the agent for it to initialise or learn well. We therefore needed to transfer more information to the agent.

Figure 6. Comparison for Bipedal environment between baseline and 1st layer transfer.

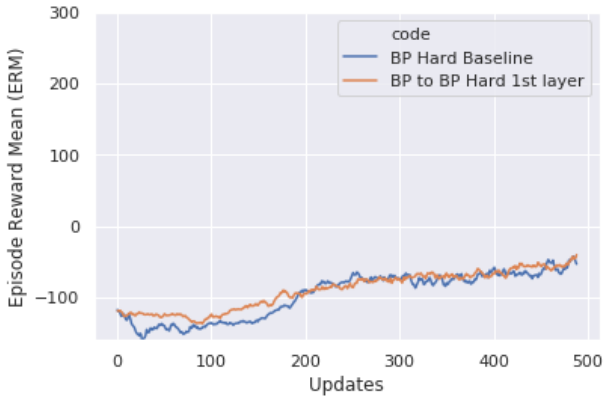

Figure 7. Comparison for Bipedal Hardcore environment between baseline and 1st layer transfer.

4.2.2. Weight sharing: all layers with 1st layer locked

We then experimented with transferring all the weights but locked the 1st layer weights, so they remained unchanged throughout training. In the first experiment we transferred the weight of *BipedalWalkerHardcore* to the *BipedalWalker* environment. Next, we shared the weights of *BipedalWalker* to the *BipedalWalkerHardcore* environment. For the first experiment, we can see the graph 8 shows us the comparison of this weight sharing model with the baseline. We can see immediately that the training time was shorter, with the model's performance starting at 180 ERM with shared weights before quickly progressing to 300 ERM. This shows that the weights trained on the more complex environment were very relevant to the simpler environment.

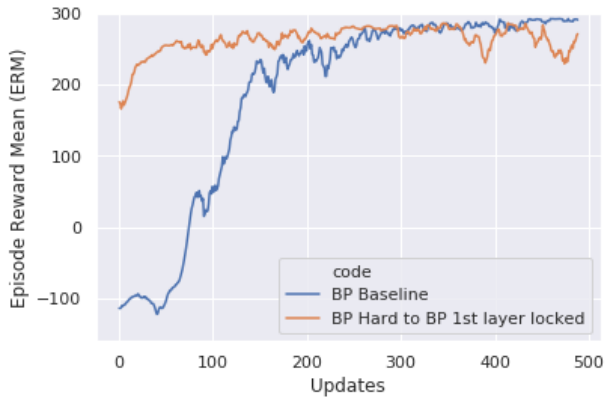

Figure 8. Comparison for Bipedal environment between baseline and all layer transfer with 1st layer locked

In the second experiment, the *BipedalWalkerHardcore* environment had not improved as much as the *BipedalWalker*. As seen in figure 9 the agent started learning at approximately -85. Though the *BipedalWalkerHardcore* showed better performance with the shared weights than the baseline through all the updates, the learning was still slow and after 500 updates the performance was comparable to the baseline. This performance is most likely caused by locking the first layer of weights before training. This will restrict the learning of the model and since it didn't begin at a high ERM, it struggled to increase it's performance.

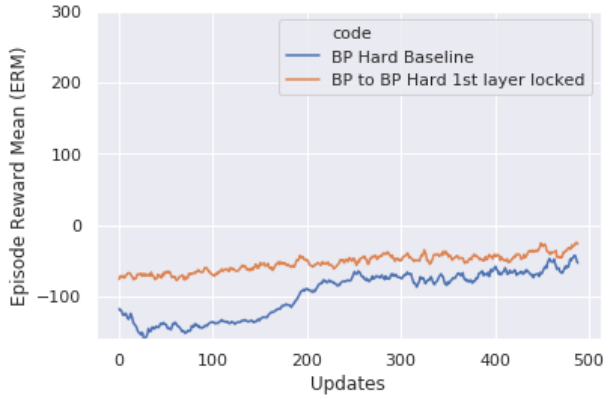

Figure 9. Comparison for Bipedal Hardcore environment between baseline and all layer transfer with 1st layer locked

4.2.3. Weight Sharing: All Weights

Finally, we tried transferring all of the weight from one environment to another. For the first experiment, we shared all the weights from *BipedalWalkerHardcore* to the *BipedalWalker* environment. Next, we experimented sharing all the weight of *BipedalWalker* to the *BipedalWalkerHardcore* environment. The weight shared includes the input, two layers and the output weights.

For the first experiment the results we got, as shown in figure 10, showed us that the agent required far fewer updates before achieving a similar maximum reward as the baseline. As seen, the ERM started at approximately 180 ERM, compared to the baseline where it started at about -110 ERM.

Figure 10. Comparison for Bipedal environment between baseline and all weights shared transfer.

This demonstrated that weight sharing was very effect in improving the initial performance and the agents policy can achieve a reasonable ERM with zero shot learning. The approach also reaches a similar performance level to training *Bipedal Walker* purely on its own but in a much faster time. A drawback is the variance from update to update for the transferred implementation, which is sporadic often achieving differences of greater than 50 after a few updates. This is likely due to the learning rate meaning that the weights of the network struggle to escape a local minimum that was reached using the transferred weights.

In the second experiment, again we transferred the weights of the input, two layers and the output from the*Bipedal Walker* to the *Bipedal Walker Hardcore*. The results of the second experiment as seen in figure 11 shows that there was a slight improvement. The hardcore agent started training at about -80 ERM when transferring all the weight of the layers.

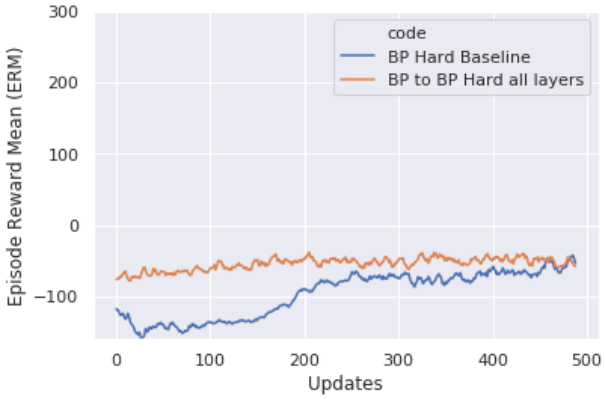

Figure 11. Comparison for Bipedal Hardcore environment between baseline and all weights shared transfer.

This demonstrated that transferring from a less complex en-

vironment to a more complex one can speed up its learning. However, the rate of improvement is low once transferred and after training both agents for 500 updates the ERM for the hardcore baseline and the transferred approach are similar. This suggests that despite improving the initial ERM score of the agent, the PPO model still struggles to learn on the more complex Bipedal Walker Hardcore environment.

4.3. Multi-task training

Our previous experiments have shown us that though transferring weights can improve the initial performance on more complex environments, the agents still struggled to learn successful policies from this point. We therefore wanted to discover if training both tasks simultaneously could improve performance by aiding the agents learning throughout the learning process. A similar method has been shown to have promising results on contious control tasks (Zhaoyang Yang, 2017).

Our agent was trained on Bipedal for one update and then on Bipedal Hardcore for one update, alternating between the two environments using the same weights.

Our results showed that the approach drastically improved the Bipedal Hardcore rate of improvement and maximum score.

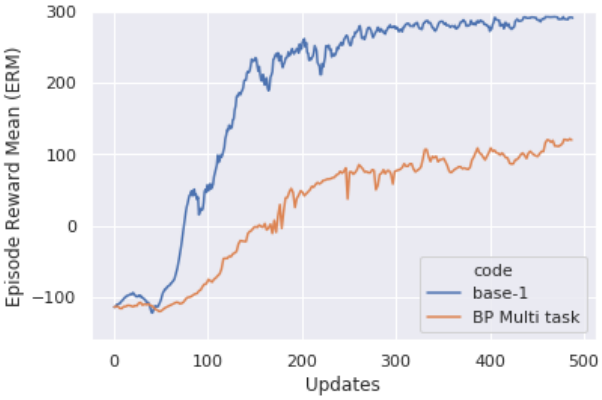

Figure 12. Comparison for Bipedal environment between baseline and multi-task. Showing only multi-task Bipedal from multi-task data.

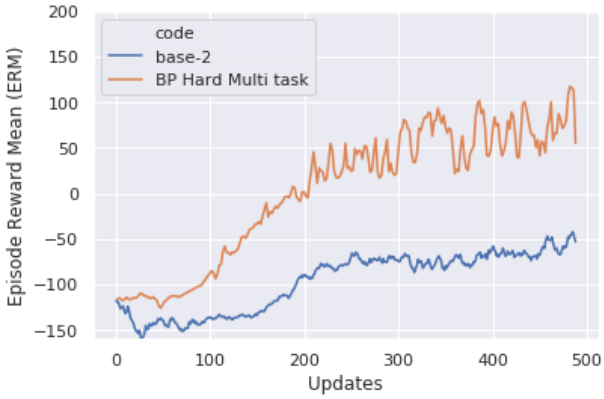

Figure 13. Comparison for Bipedal Hardcore environment between baseline and multi-task. Showing only multi-task Bipedal Hardcore from multi-task data.

This indicates the easier environment aided in the harder environments development. This could be because the easier environment acts as a stabiliser to learn an effective policy. The alternation could also ensure the policy can generalise to new instances, so when in the hardcore environment randomly places obstacles and jumps to be overcome then the agent is more resilient to changes.

5. Related work

Recent work on transfer learning in reinforcement learning was different from one study to another. Some of the related works were the following:

- Atkeson & Santamaria (1997) transfer between tasks in which only the reward function can differ are again considered. Their method successfully transfers a locally weighted regression model of the transition function, which is learned in a source task, by directly applying it to a target task. Because their model enables planning over the transition function and does not account for the reward function, they show significant improvement to the jump start and total reward, as well as the asymptotic performance.
- Higgins et al. (2017) used an unsupervised vision objective to produce robust features for a policy, and found that this policy was able to transfer to previously unseen vision tasks in DeepMind[7] Lab and MuJoCo [8].
- Mehta et al. (2008) assumes that the learner will train on a sequence of tasks which are identical except for different reward weights. The re- ward weights define how much reward is assigned via a linear combination of reward features. The authors provide the reward features to the agent for a given set of tasks.
- Parisotto et al. (2015) enable transfer to a new task, they first remove the final softmax layer of the AMN.

[7]https://deepmind.com/blog/open-sourcing-deepmind-lab/
[8]http://www.mujoco.org

Then used the weights of AMN as an instantiation for a DQN that will be trained on the new target task.

- Ha (2018) have trained the *BipedalWalker* environments using PPO and TRPO and it work well when the agent is presented with a well-designed dense reward signal, while population-based methods offer computational advantages for sparse-reward problems.

We have transferred the knowledge by the weight sharing as described in details in section 3.1 between the tasks which is similar to what Atkeson & Santamaria (1997) and Mehta et al. (2008) have done. The PPO was proven to be the best to train the agent by Ha (2018) and that helped us to make the decision in using this method for the experiments we have done in section 4.

6. Conclusion

To sum up our report, we have used the *BipedalWalker-v2* and *BipedalWalkerHardcore-v2* environments of the OpenAi gym and explained them in detail. Then, we introduced the methodology of reinforcement learning, PPO and transfer learning techniques that we used to train our agents. Next, we started our experiments with the baselines followed by the transfer learning experiments which were using weight sharing and multi-task training techniques.

Our results showed us that though weight sharing could increase the zero-shot performance of our agents, the more complex environments still posed a challenge for our models. However, we then found that multi-task training greatly improved our performance on the more complex environments, eventually achieving our highest reward on the Bipedal Walker Hardcore environment.

This answers our question by showing that we can improve the performance of reinforcement learning algorithms on continuous control environments through the use of transfer learning, especially multi-task learning.

7. Future work

For our future work, we would try applying the transfer learning on Mujoco environments such as the Hands environment which are based on the Shadow Dexterous Hand. By introducing transfer learning to the best performing models, we would aim to improve performance on hand manipulation tasks as well as reducing the time and cost of training.

Moreover, we would like to investigate combinations of multi-task and weight sharing to investigate if further gains could be made.

References

Atkeson, Christopher G and Santamaria, Juan Carlos. A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pp. 3557–3564. IEEE, 1997.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Catto, Erin. Box2d v2.3.0 user manual. 2013.

Ciosek, Kamil Andrzej and Whiteson, Shimon. Offer: Off-environment reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Dhariwal, P., Hesse C. Klimov O. Nichol A. Plappert M. Radford A. Schulman J. Sidor S. and Wu, Y. Openai baselines. *GitHub, GitHub repository*, 2017.

Ha, David. Reinforcement learning for improving agent design. *arXiv preprint arXiv:1810.03779*, 2018.

Higgins, Irina, Pal, Arka, Rusu, Andrei, Matthey, Loic, Burgess, Christopher, Pritzel, Alexander, Botvinick, Matthew, Blundell, Charles, and Lerchner, Alexander. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1480–1490. JMLR. org, 2017.

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

Kaelbling, Leslie Pack, Littman, Michael L, and Moore, Andrew W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Kalashnikov, Dmitry, Irpan, Alex, Pastor, Peter, Ibarz, Julian, Herzog, Alexander, Jang, Eric, Quillen, Deirdre, Holly, Ethan, Kalakrishnan, Mrinal, Vanhoucke, Vincent, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kojcev, Risto, Etxezarreta, Nora, Hernández, Alejandro, and Mayoral, Víctor. Evaluation of deep reinforcement learning methods for modular robots. *arXiv preprint arXiv:1802.02395*, 2018.

Konidaris, George and Barto, Andrew. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 489–496. ACM, 2006.

Mehta, Neville, Natarajan, Sriraam, Tadepalli, Prasad, and Fern, Alan. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289, 2008.

Parisotto, Emilio, Ba, Jimmy Lei, and Salakhutdinov, Ruslan. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

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

Tirinzoni, Andrea, Sessa, Andrea, Pirotta, Matteo, and Restelli, Marcello. Importance weighted transfer of samples in reinforcement learning. *arXiv preprint arXiv:1805.10886*, 2018.

Zhang, Amy, Ballas, Nicolas, and Pineau, Joelle. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Zhang, Shangtong and Zaiane, Osmar R. Comparing deep reinforcement learning and evolutionary methods in continuous control. *arXiv preprint arXiv:1712.00006*, 2017.

Zhaoyang Yang, Kathryn Merrick, Hussein Abbass Lianwen Jin. Multi-task deep reinforcement learning for continuous action control. 2017.