Transfer Learning for Deep Reinforcement Learning Agents

Trevor Barron, Matthew Whitehead, and Alan Yeung

Colorado College, Colorado Springs CO 80903, USA
{trevor.barron, matthew.whitehead, alan.yeung}@coloradocollege.edu

Abstract. Researchers are using deep neural networks and reinforcement learning to train agents to perform complex tasks at unprecidented levels, even better than humans in many cases. However, there are still tasks that are too computationally intense for researchers to practically pursue with the current methods. Thus, we explore a method to augment deep reinforcement learning that has the potential to speed up learning time and improve overall performance, transfer learning. Specifically, we use a human guided transfer learning, where the source tasks are programmed by a human. We find that transfer learning provides a substantial learning boost to the agent in the task we have chosen, and provide evidence that the potential benefits of transfer learning, particularly human guided transfer learning, may be worth the costs.

Keywords: deep learning, reinforcement learning, transfer learning, convolutional neural networks

1 Introduction

Deep reinforcement learning (DRL) is currently being used by researchers to create agents that are out performing any that have come before in artificial intelligence (AI) and machine learning (ML) tasks. For instance, in [1] and [2] researchers were able to utilize DRL to create agents that outperformed human experts in several Atari 2600 games and the extremely complex game of Go respectively. These are unprecedented achievements and have been important milestones in the advancement of artificial intelligence and machine learning. However, even with the use of DRL, there are still tasks that are too computationally intense or time consuming for most researchers to practically pursue.

Thus, we explore the use of transfer learning (TL) as an augmentation to DRL in an attempt to speed up learning. Other research has provided evidence that reinforcement learning (RL) and TL used together can significantly speed up an agent's learning. Many works focus on developing and analyzing the effects of new TL algorithms on RL agents [3, 4, 5]. Some works, such as [6], focus on developing techniques to transform the action value function so that it can be directly applied to new target tasks. The authors develop a TL method and run analysis similar to ours, comparing the improvement of agents that use TL (TL agents) and agents that do not use TL (non-TL agents). However, few papers have analyzed the learning speed up of DRL combined with TL, which may provide even better speedup in learning compared to RL and TL.

In this paper we focus on analyzing both the performance and learning time benefits of using DRL in conjunction with TL. We train an agent to perform a simple task in a Minecraft like environment, and show that the combined use of DRL and TL provide a substantial learning speedup. The DRL network from [1] was used in conjunction with human programmed environments for the TL. Although there are considerable drawbacks and limitations to using human selected source tasks for TL, we believe that these findings show that the potential benefits of DRL and TL combined are significant enough as to outweigh these limitations in many AI and ML tasks.

2 Background

We use the same deep convolutional neural network (DCNN) and reinforcement learning (RL) algorithm outlined in [1] for our experiments. The network structure is as follows: the input to the DCNN consists of an 84 X 84 X 4 image, the first hidden layer convolves 16 8 X 8 filters with stride 4 with the input image and applies a rectifier non-linearity, the second hidden layer convolves 32 4 X 4 filters with stride 2 and applies another rectifier non-linearity, the final hidden layer consists of 256 fully connected rectifier units, and finally the output layer is a fully connected linear layer with an output for each valid action. As in [1] we train our network using only pixel input and the rewards recieved from the actions performed.

The RL algorithm is a variation of Q-learning where the goal is to generate a function that approximates the policy $Q^*(s,a)$, the optimal action-value function, which is defined as the maximum achievable return from following any strategy after seeing sequence s and performing some action a, $Q^*(s,a) = max_{\pi}\mathbf{E}_{s'\sim\epsilon}[r + \gamma max_{a'} Q^*(s',a')|s,a]$. That is, $Q^*(s,a)$ is a function that maximizes return (R_t), that is defined as the reward from the current action (r) plus the stream of future rewards till termination (T) discounted by γ at each time step t, $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. In order to approximate $Q^*(s,a)$, a DCNN is used as our function approximator with weights set to θ in iteration i, $Q(s,a,\theta_i)$.

In order to have the DCNN approximate $Q^*(s,a)$ the DCNN is trained with the goal of minimizing a sequence of loss functions, $L_i(\theta_i) = \mathbf{E}_{s,a\sim p(\cdot)}[(y_i - Q(s,a,\theta_i))^2]$ that change at each iteration i, where $y_i = \mathbf{E}_{s'\sim\epsilon}[r+\gamma max_{a'} Q(s',a';\theta_{i-1})|s,a]$ is the target for iteration i and $p(s,a)$ is a probability distribution over sequence s and actions a. By updating weights according to the goal of minimizing $L_i(\theta_i)$, $Q(s,a,\theta_i)$ can be shown to approach $Q^*(s,a)$ given sufficient training time. For more details on the DCNN structure or the RL algorithm see [1].

The term transfer learning (TL), also called inductive transfer, refers to the transfer of knowledge learned in different, but related contexts [5]. Examples of transfer learning abound in real life, such as learning to ride a bike after learning to ride a tricycle or learning to run after learning to walk.

In a machine learning context, researchers studying TL are concerned with the added benefit that learning one task has on learning another (usually related) task. Often times TL is used to speed up an agents learning on a desired task, known as the target task, by first training the agent on a source task, before transferring the agent to continue training on the target task. The hope is that knowledge from the source task will be transferred to the target task and the learning time necessary for the agent to achieve a threshold performance will be decreased.

There are various measures for the benefits of TL [7], but for our purposes the total time measure is used. That is, the benefit of TL is measured by the difference in total training time, including training on source tasks, for some threshold performance measure.

In this paper we use a human programmed version of TL, where the source tasks are preselected by the human, and a make use of a DCNN as our function approximator for the RL algorithm. The agents (one using TL and one not) perform a simple task described in section 3.1 and the results from the experiments are described in section 4.

3 Experiments

In order to analyze the potential benefits of TL, we trained agents augmented with TL and compared their learning to agents not augmented with TL. If TL provides learning benefits to agents, then we would expect to see a quicker rate of learning for agents augmented with TL. That is, agents trained with TL should achieve the same reward on a specific task in less time than agents trained without TL. If there is no drastic difference in learning times, or a negligible difference, then there are two possibilities: TL may not be beneficial for the target task or the source tasks selected may not stimulate TL.

We tested our environment with both the Torch and Caffe deep learning frameworks [8, 9]. For all of our experiments and results we used the Caffe framework as it was more efficient for our purposes compared to Torch.

3.1 Setup

Our experiments use a simple goal oriented task where the goal is to maximize the reward earned. The target task environment is depicted in figure 1. The agent earns one point of reward for going further in the Z (forward) direction than ever previously (let's call this Z_max), zero points for being at a Z position where $Z = Z\_max$, and negative one point if in a position $Z < Z\_max$. The negative rewards were meant to increase the efficiency of the agent by decreasing inefficient actions by the agent, such as moving backwards, forwards, backwards, etc. The reward of the target task is maximized by creating a bridge of six blocks in size and walking forward to the end of the walkway (figure 2).

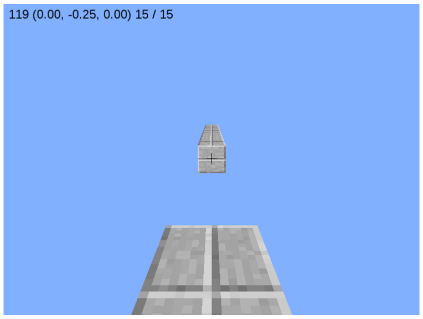

Fig. 1: Target task

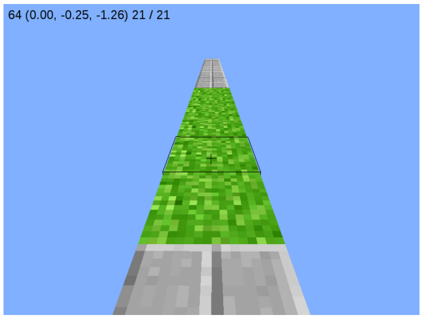

Fig. 2: Created bridge

A target task with a gap of six blocks was chosen because that is the maximum number of blocks that can be removed that still allows a bridge to be created.

To augment agents with TL we created five source tasks to train the agent for the target task.

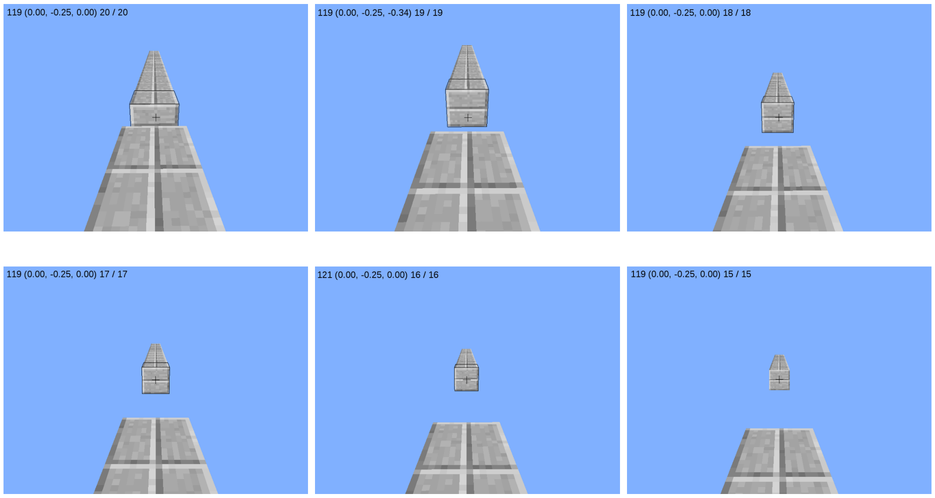

Fig. 3: Progression of TL source tasks from the simplest (walkway with gap size 1) to the target task (walkway with gap size 6).

The source tasks feature a gap ranging from size one block to five blocks in the pathway. The agent starts on the simplest source task, the environment with a gap of one block, and progresses to the most complex source task, the environment with a gap of five blocks, before advancing to the target task, the environment with a gap of six blocks. By training the agent on simpler source tasks than the target task, the hope is that the agent will learn the optimal policy, creating a bridge, faster than the agents that begin training directly on the target task.

The TL agent is transitioned to the next most complex source task based on performance. If the agents average reward from the last ten games is above a certain threshold, then the agent transitions to the next source task, or the target task, if it was on the most complex source task. The threshold used is the minimum amount of reward necessary for the agent to have created a bridge. Specifically, the maximum reward attainable without a bridge is roughly twelve points (3 blocks * 4 points per block), plus a few extra possible points if the agent falls forwords off the edge, thus we use an average threshold score of sixteen over the last ten games. If the agent recieves an average score of at least sixteen, the agent must have created a bridge successfully at least one time. Generally, a ten game average score of at least sixteen indicates that the agent can create bridges somewhat consistently because without a bridge an untrained agent usually garners negative points, with inefficient movements described above. The agents have seven possible actions, move forewords, move backwards, rotate up, rotate down, create a grass block, destroy a grass block, or do nothing. These actions were selected because they are the minimum actions needed to create a bridge and achieve a maximum reward, any other actions would be learned away and increase the total training time for both the TL and non-TL agents. There are two possible ways for an episode to end, if the agent steps off the bridge, either by walking forwards or backwards off an edge, or if the maximum frame limit per round (500 frames) is reached, the episode ends and a new episode begins with the agent in the starting position.

As discussed in [1], the agent sees and selects actions every fourth frame, which roughly increases the number of games the agent can play by four times, because it takes substantially less computation to run the episode forward one frame than to have the agent select a new action. On every skipped frame the agent's action is repeated, with the exception of break a grass block and create a grass block, which are only performed once every four frames in order to prevent the agent from creating or breaking four blocks inadvertently. Each episode lasts for a maximum of 500 frames and each epoch is 5000 frames. However, this does not necessarily translate perfectly into ten episodes per epoch because (as often happens) the agent can die by walking off the path.

4 Results and Discussion

Our results provide evidence that shows TL to be an effective tool in speeding up learning, particularly with a task that includes a delayed reward. Figure 4 and figure 5 show the average game reward over the number of training epochs for two trials. The approximate maximum reward achievable (roughly eighty-four points) is represented by the black line. We use the term “approximate” because it is possible for an agent to achieve slightly more than eighty-four points by falling forward off the edge at the end of the walkway. The red line represents whether or not the agent is able to consistently create a bridge in the target task. As discussed in section 3.1, the approximate minimum number of points achievable after creating a bridge is sixteen points.

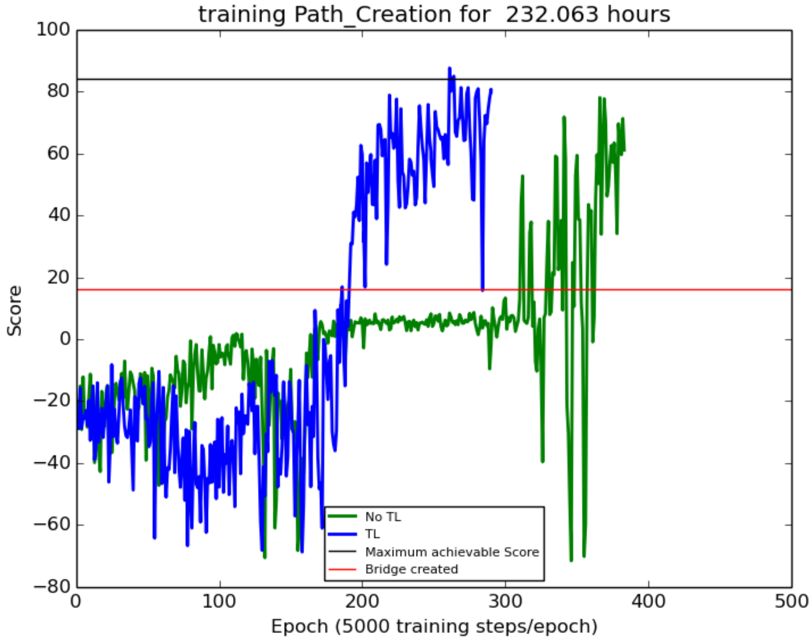

Fig. 4: TL agent Vs. non-TL agent trial one

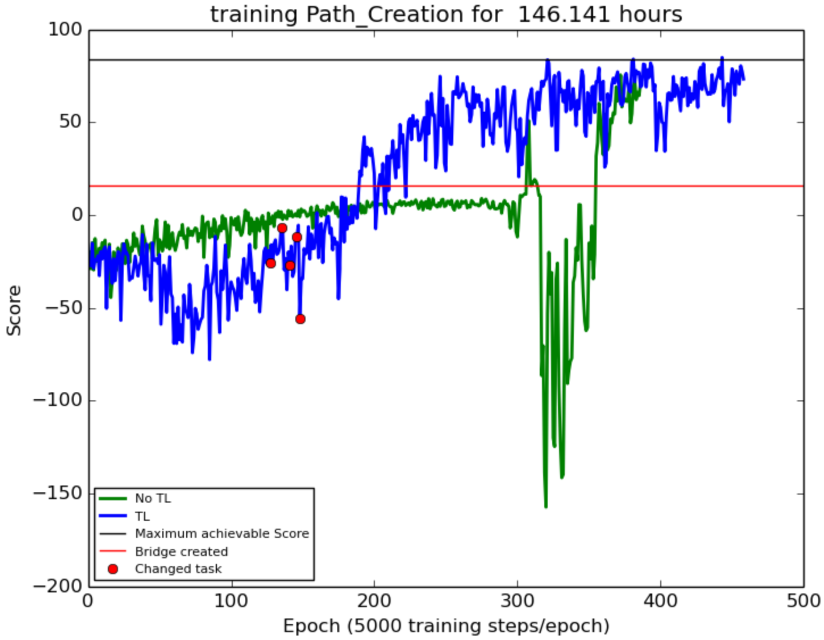

Fig. 5: TL agent Vs. non-TL agent trial two with red dots indicating a change in source task

The red points in figure 5 represent the approximate epoch (since task transitions usually occur between epochs) where the agent's source task changed. The first red point represents the transition from the source task of gap size one to the source task of gap size two, similarly for the second, third, and fourth red dot, while the fifth red dot represents the transition from the the source task of gap size five to the target task of gap size six.

Figure 4 and figure 5 depict a clear learning advantage later on for the TL agent compared to the non-TL agent. Both graphs show the TL agent underperforming the non-TL agent in terms of reward early on. This difference in rewards early on may be attributable to the difference in tasks for the TL agent compared to the non-TL agent, particularly it may have been challenging for the TL agent to generalize the lessons learned from source tasks to new tasks.

However, once the TL agent transitions to the target task, the average game reward (which is an indicator of learning) steadily increases. Specifically, when looking at figure 5 after training on all source tasks, the TL agent vastly out performs the non-TL agent in terms of average reward achieved. It is interesting to observe that as the average game reward of the TL agent increases at a steady rate (after reaching the target task), the average game reward of the non-TL agent appears to level off for a significant time (roughly one hundred epochs). The average reward of the non-TL agent levels off around the maximum achievable reward without creating a bridge (about twelve points), which indicates that the non-TL agent was having a difficult time discovering the optimal policy, creating a bridge. In order for the non-TL agent to have any notion of the benefits of creating a bridge, the non-TL agent must first choose (often randomly) to place six blocks to cover the gap, and then walk across. That is, the reward is delayed until several "correct" actions are performed. As the graphs indicate, this was quite a time consuming challenge for the non-TL agent. On the other hand, the TL agent was able to grasp the vital concept of creating a bridge early on as it trained on each source task. Once the TL agent transitioned to the target task it simply needed to apply its previous learning to create a bridge and learn efficient movements to accumulate reward.

Table 1 displays the number of epochs needed for each agent to consistently create a bridge and achieve the maximum achievable reward respectively (where the former must be done before the latter). Since the training epochs for non-TL and TL agents take the exact same time, the number of epochs also represent the total amount of time an agent took to learn the goal.

Table 1 shows that the TL agent was able to consistently build a bridge in thirty-nine percent less time, on average, compared to the non-TL agent. Additionally, the table indicates that the TL agent was able to earn the maximum achievable reward consistently while the non-TL agent was unable to earn the maximum achievable reward during the allotted training time. It is important to note that the time taken by the TL agent to earn the maximum achievable reward is not consistent (261 epochs in trial one vs. 381 epochs in trial two), however given that the non-TL agent was never able to achieve a similar feat, the

Table 1: Epochs taken to achieve goal

	Trail 1		Trail 2	
	Built Bridge	Max Achievable Reward	Built Bridge	Max Achievable Reward
TL agent	186	261	190	381
Non-TL agent	311	N/A	307	N/A
Percent change	-40%	N/A	-38%	N/A

Note: The table shows the number of epochs (time) taken by each agent to build a bridge and achieve the maximum achievable score in the target task, where percent change= (TL epochs−non-TL epochs)/non-TL epochs.

inconsistency may not be a large concern. These results indicate, quite clearly, the advantages in total training time when using TL methods.

These results do indicate that TL combined with RL has the potential to substantially speed up learning. However, it is important to acknowledge the limitations of our use of TL, the first and foremost being that source tasks were selected by humans. To date, there is no program that uses fully automated TL, that is no program can successfully select an appropriate source task from a target task, learn how the source and target tasks are related, and effectively transfer knowledge from the source task to the target task, and our methods are no exception [7]. The performance benefits of TL undeniably depend heavily on the source tasks selected, and since our TL methods use human selected/programmed source tasks our results may vary drastically with different source and target tasks. Another limitation of TL in general is that there is no set method to choose source tasks that guarantees TL will occur. In order for TL to occur, the source and target tasks must be similar enough that an agent can generalize learning, but different enough that the source task helps in the learning of the target task. Currently, there is no measure for the similarity of source and target tasks that guarantees the effectiveness of the source task in TL [7]. Additionally, the performance benefits of TL may vary from trial to trial, as the difference in the number of epochs taken to acheive the maixmum achievable reward by the TL agent (see table 1) indicate. Thus, as with many TL methods, there are no guarantees as to how much better a TL agent will perform compared to a non-TL agent, as the algorithms we use have an element of randomness.

Even with these limitations, our results indicate that TL may provide substantial benefits to RL that are worth the costs. An average learning speed up of 39% (or even 10%) is quite substantial when it takes days to train an agent. Thus, even if there is the possibility that TL may provide learning speed up, it is worthwhile for researchers to consider human guided TL as a method to increase learning and potential performance.

5 Future Work

Fortunately, the limitations of our TL methods provide promising avenues for future research. There is some work being done on automating TL source selection, but full automation may still be a while away. However, creating a semi automated TL algorithm seems like a particularly promising research avenue. It may be possible to train an agent to act like a coach and select source tasks, from a set (or sets) of human provided tasks, to train an agent effectively. Depending on how the sets of tasks are defined, the “coach” may even be somewhat generalizble and may be able to train agents on other tasks with a different set of provided source tasks.

Additionally, it may be possible to train a DNN to approximate the similarity of a source task and target task and aid in TL training of agents. If a DNN could approximate the similarity of a source task and target task to an accurate (for some definition of acurate) enough degree, then we would be one step closer to fully automating TL. Finally, more work needs to be done in measuring the benefit of TL on different types of tasks and different selections of source tasks. Does TL work better with delayed reward tasks or immediate reward tasks? Would a different combination of source tasks than the ones we chose, such as skipping gaps of even length, lead to faster or slower learning? More work needs to be done to answer these and many other important questions in the domain of TL combined with DRL.

Bibliography

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[3] Matthew E Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pages 879–886. ACM, 2007.

[4] Matthew E Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 283–290. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[5] Jan Ramon, Kurt Driessens, and Tom Croonenborghs. Transfer learning in reinforcement learning problems through partial policy recycling. In *Machine Learning: ECML 2007*, pages 699–707. Springer, 2007.

[6] Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via intertask mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[7] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

[8] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.