An Awesome Title

Alan Yeung[*]
Senior Capstone
Department of Computer Science
Colorado College
902 N Cascade Ave, Colorado Springs, CO 80903
`Alan.Yeung@coloradocollege.edu`

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Deep reinforcement learning (DRL) is currently being used by researchers to create agents that out perform most others that have come before in artificail intelligence (AI) and machine learning (ML) tasks. For instance, in [1] and GO researchers were able to utilize (DRL) to create agents that outperformed human experts in several Atari 2600 games and Go respectively. These are unprecidented achievements and have been important milestones in the advancement of artificail 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 (CITE). 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 (CITE) was used in conjuction with human programmed environments for the TL. Although there are considerable drawbacks and limitations to using human programmed environments for TL, we believe that these findings show that the benefits of DRL and TL combined are significant enough as to outweight these limitations in many AI and ML tasks.

2 Background

We use the same deep neural network (DNN) and reinforcement learning (RL) algorithm outlined in (CITE) for our experiments. As described in (CITE) the input to the DNN 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 nonlinearity, the second hidden layer convolves 32 4 X 4 filters with stride 2 and applies another rectifier nonlinearity, the final hidden layer consists of 256 rectifier units and is fully connected, and finally the output layer is fully connected linear layer with an output for each valid action.

[*]`http://yeungalan0.github.io/`

Additionally, the RL algorithm uses a return that is the reward from the current action (r) plus the stream of future rewards till termination (T) discounted by γ at each time step t, $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. The optimal action-value function $Q^*(s,a)$ is defined as the maximum achievable return from following any strategy after seeing sequence s and performing some action a, $Q^*(s,a) = max_\pi \mathbf{E}_{\mathbf{s'}\sim\epsilon}[\mathbf{r} + \gamma \overset{\mathbf{max}}{\mathbf{a'}} \mathbf{Q}^*(\mathbf{s'},\mathbf{a'})|\mathbf{s},\mathbf{a}]$. $Q(s,a,\theta_i)$ is a neural network with weights set to θ in interation i. The goal is to minimize a sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i, $L_i(\theta_i) = \mathbf{E}_{\mathbf{s},\mathbf{a}\sim\mathbf{p}(\cdot)}[(\mathbf{y_i} - \mathbf{Q}(\mathbf{s},\mathbf{a},\theta_\mathbf{i}))^\mathbf{2}]$. $y_i = \mathbf{E}_{\mathbf{s'}\sim\epsilon}[\mathbf{r} + \gamma \overset{\mathbf{max}}{\mathbf{a'}} \mathbf{Q}(\mathbf{s'},\mathbf{a'};\theta_{\mathbf{i-1}})|\mathbf{s},\mathbf{a}]$ is the target for iteration i and $p(s,a)$ is a probability distribution over sequence s and actions a. For more details on the DNN structure or 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 [2]. Examples of transfer learning abound in real life, such as learning to ride a bike after learning to ride a tricycle or learning to walk after learning to run. 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. There are various measures for the benefits of TL [3], 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 for the similar performance of two agents, one augmented with TL and one not.

In this paper we use a human programmed version of TL, where the source tasks are preselected by the human, and a DNN. The agents (one using TL and one not) perform a simple task described in section 3 (Experimental Setup) and the results from the experiments are described in section 4 (Results).

Other studies have found that RL coupled with TL con produce a speedup in learning (CITE).

3 Experimental Setup

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 benefit learning or the source tasks selected may not stimulate TL.

We tested our environment with both the Torch and Caffe deeplearning frameworks [4, 5]. For all of our experiments and results we used the Caffe framework as it decreased training time compared to Torch.

3.1 Setup

For our experiments we use a simple goal oriented task where the goal is to maximize the reward earned. The target task environment is depicted below. 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 3.1).

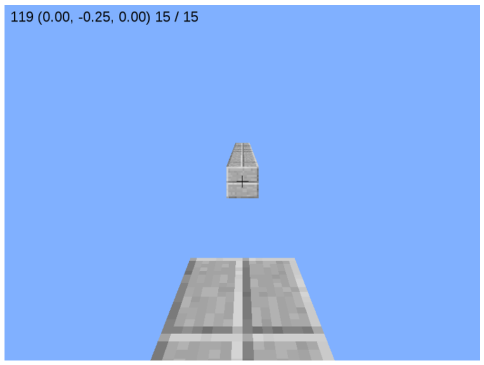

Figure 1: Target task

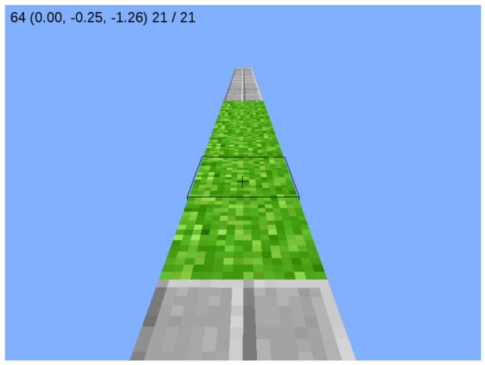

Figure 2: Created bridge

A target task of 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.

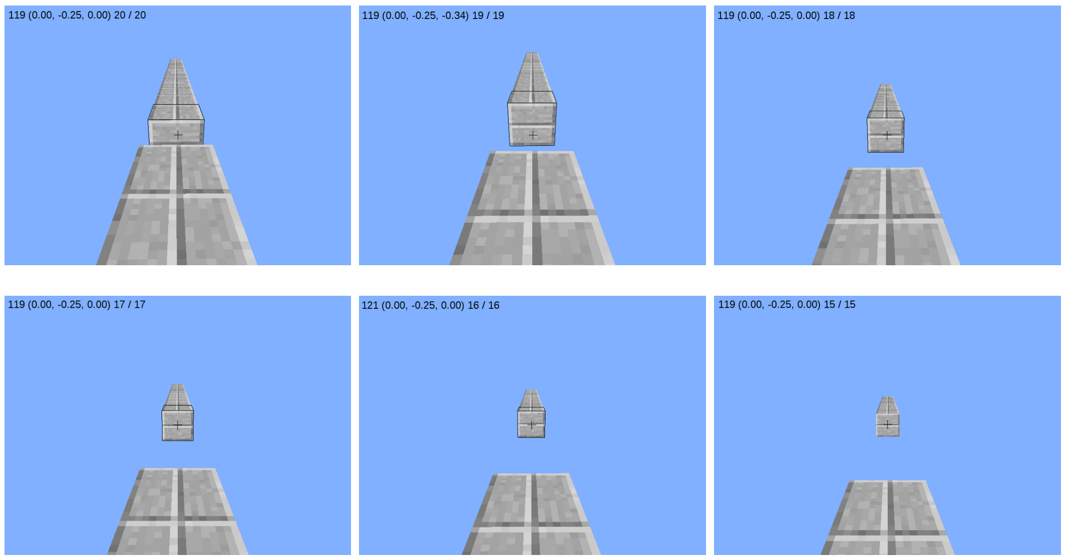

Figure 3: Progression of transfer learning source tasks from the simplest (walkway with gap size 1) to the target task (walkway with gap size 6).

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 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 amount of minimum reward necessary for the agent to have created a bridge. Specifically, the maximum reward attainable without a bridge is about twelve points (3 blocks * 4 points per block), thus if the agent achieves an average score of at least fifteen over the last ten games, the agent must have created a bridge successfully at least one time. Generally, a ten game average score of at least fifteen 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 forewards, 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 minimal 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 a round to end, if the agent steps off the bridge, either by walking forwards or backwards off an edge, or if the maximum frame limits per round is reached, the round ends and a new round begins with the agent in the starting position.

4 Results

Include graph of results.

What are the results/interpretations? Add your graph results here. Interpret the results, why, cause, limitations, etc Idea is similar to leveling up in video games, you become better at some task with each level up, but you might not have mastered any task at a specific level.

5 Conclusion

Conclude and suggest future avenues for research!

References

[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] 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.

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

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

[5] 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.