Initial Progress in Transfer for Deep Reinforcement Learning Algorithms

Yunshu Du*, Gabriel V. de la Cruz*, James Irwin, and Matthew E. Taylor
School of Electrical Engineering and Computer Science
Washington State University, Pullman, Washington
{yunshu.du,gabriel.delacruz,james.irwin}@wsu.edu, taylorm@eecs.wsu.edu

Abstract

As one of the first successful models that combines reinforcement learning technique with deep neural networks, the Deep Q-network (DQN) algorithm has gained attention as it bridges the gap between high-dimensional sensor inputs and autonomous agent learning. However, one main drawback of DQN is the long training time required to train a single task. This work aims to leverage transfer learning (TL) techniques to speed up learning in DQN. We applied this technique in two domains, Atari games and cart-pole, and show that TL can improve DQN's performance on both tasks without altering the network structure.

1 Introduction and Related Work

The recent development of the Deep Q-network (DQN) [Mnih *et al.*, 2015] algorithm has attracted a lot of attention with its ability to master 49 different Atari games with little prior knowledge. The combination of Reinforcement Learning (RL) [Sutton and Barto, 1998] with a deep Convolutional Neural Network (CNN) as a function approximator has allowed agents to learn directly from raw image inputs. Although DQN performed well in many different game domains with the same architecture, one limitation of this approach is that the training time to train a single task is prohibitively long.

Many attempts have been made to address this problem. [Nair *et al.*, 2015] proposed a massive distributed architecture for DQN that significantly reduced training time and achieved a new state-of-the-art result in the Atari domain. [Liang *et al.*, 2015] performed systematic analyses in DQN and were able to capture some of the key features from DQN to a linear representation, thus reducing the burden of learning that resulted in a simpler model that can achieve near-expert performance with lower computational requirements. Policy distillation [Rusu *et al.*, 2015] and actor-mimic network (AMN) [Parisotto *et al.*, 2015] deployed model compression techniques [Bucilu *et al.*, 2006] to DQN and built a multitask agent which can play Atari games nearly as good as a single task DQN, without increasing the model complexity. In addition, the learned AMN was able to transfer its multitask knowledge back to the original DQN and showed positive effects in the learning speed of DQN in some of the Atari games.

However, the above methods either rely on hardware acceleration or require separate model training. In this paper, we show that it is possible to leverage transfer learning [Taylor and Stone, 2009] techniques to directly improve learning speeds for DQN in playing Atari games. Our results show that knowledge transfer between novel but related game domains can improve learning speed as well as accumulated reward. In addition, we applied DQN to a simulated cart-pole balancing task to show the possibility of transfer in DQN for physical robots. The same transfer methods were performed and also have successfully sped up the learning of the cart-pole task.

This paper is structured as follows: in the next section we briefly review the Deep Q-network algorithm and the concept of transfer learning. Section 3 describes our experimental setup for both Atari games and cart-pole tasks and presents results, followed by discussions in Section 4. An outline of future work concludes the paper.

2 Background

In this section we briefly review the Deep Q-Network algorithm and the concept of transfer learning.

2.1 Deep Q-Network

A reinforcement learning (RL) agent aims to learn to interact with an unknown *environment* such that the cumulative *reward r* is maximized. Such an environment can be represented as a Markov Decision Process (MDP), described by a 5-tuple $\langle S, A, P, R, \gamma \rangle$. At timestep t, an agent observes a state $s_t \in S$, chooses an action $a_t \in A$ which returns reward $r \sim R(s_t, a_t)$ and leads to next state $s_{t+1} \sim P(s_t, a_t)$. The discount factor γ determines the importance between future rewards and immediate rewards.

The Deep Q-network (DQN) algorithm combines Q-learning [Watkins and Dayan, 1992] with a deep convolutional neural network (CNN) to learn a generalized representation of different Atari game environments directly from observation of game screens. Similar to classic Q-learning, the agent aims to select actions that maximize cumulative reward by approximating the optimal Q function.

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

*The first two authors contributed equally to this paper.

The DQN algorithm uses a CNN to approximate the Q function, $Q(s, a; \theta) \approx Q^*(s, a)$, where θ is the network's weight parameters. For each iteration i, DQN was trained to minimize the *mean-squared error* between the Q-network and its target $y = r + \gamma max_{a'} Q(s', a'; \theta_i^-)$, where θ_i^- is the weight parameters for the target Q-network that was generated from previous iterations. The loss function can be expressed as:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y - Q(s, a; \theta_i))^2] \quad (2)$$

where $\{s, a, r, s'\}$ are state-action samples drawn from *experience replay* memory [Mnih *et al.*, 2015]. The use of experience replay, along with a *target network* and *reward clipping* help to stabilize learning.

2.2 Transfer Learning

Transfer learning (TL) is a machine learning technique that aims to improve the learning speed on one or more *target tasks* by making use of the knowledge learned from one or more *source tasks* [Taylor and Stone, 2009]. The *source task selection* and *task difference assumptions* are two vital components that can decide whether TL will be beneficial. TL assumes a human has performed source task selection thus we hand-pick source tasks for both the Atari and cart-pole domains in this work. The intuition of TL is that generalization not only exists within a single task but also across domains, even if agents use different state representations or action spaces. In the context of deep learning, we expect that the features learned in one task (e.g., the first level of the CNN) will be useful for multiple types of tasks. In the Atari domain, we select the games of *Breakout* and *Pong* as our source/target task pair because of their shared similarity (see section 3.1). This is considered as *far-transfer*. In the cart-pole task, we focus on *near-transfer*, that is transfer within a single domain but with different system dynamics (see section 3.2).

We evaluate TL via four metrics [Taylor and Stone, 2007]: 1) *jumpstart*: the agent's initial performance on the target task was improved by transferring source task knowledge, 2) *final performance*: the agent's final performance on the target task was improved via transfer, 3) *total reward*: the accumulated reward on the target task was improved compared to no-transfer learning (within the same learning time period), and 4) *transfer ratio*: the ratio of total reward obtained by the transfer agent and the non-transfer agent.

Studies of TL in deep learning for image classification have achieved positive results [Razavian *et al.*, 2014; Girshick, 2015]. Here we apply a similar strategy — we leverage one particular existing technique [Yosinski *et al.*, 2014] and focus on investigating transferable features in the three convolutional layers of DQN for both of our tasks.

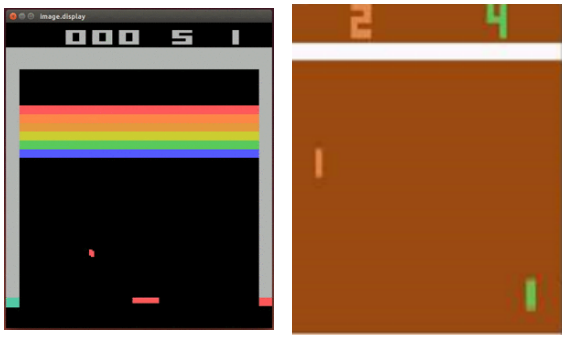
3 Experiments and Results

We apply TL to the DQN algorithm for two different tasks: Atari game playing and cart-pole balancing. For the Atari tasks, we use the existing Google DeepMind source code [Mnih *et al.*, 2015]. For the cart-pole tasks, we developed a cart-pole simulator using *pygame* [Shinners, 2011] and we are using a different implementation of DQN[1] that uses *Tensorflow* [Abadi *et al.*, 2015]. Both the DeepMind and Tensorflow code are modified to support weight transfer.

3.1 Atari Game

The Arcade Learning Environment (ALE) [Bellemare *et al.*, 2013] is an RL platform that interfaces with over 500 *Atari 2600* games. Atari was designed to be challenging even for human players and the variety of environments has made it difficult to successfully generalize across games.

We first consider the selection of the source/target task pair. As mentioned in section 2.2, similar games should be selected. Here we define the similarity between two games based on whether they are the same type (action game, strategy game, racing game, etc.) and/or if they have similar valid actions (Fire, North, South, West, East, etc.). We pick the games of Breakout and Pong as our tasks (Figure 1) because both games are action games that try to hit a moving ball with a paddle, and they have four and three valid game actions, respectively. In particular, Breakout has a Fire action at the beginning of the game while Pong does not, but the other three actions are the same.



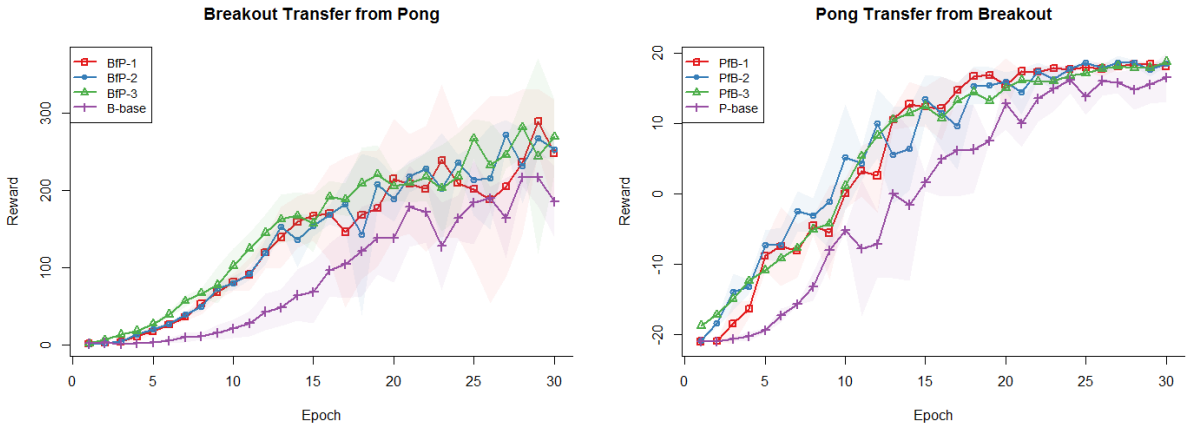
(a) Game of Breakout (b) Game of Pong

Figure 1: Selection of source/target task pair in Atari: we pick the games of Breakout and Pong because they consist of similar game environments (i.e., the agent is trying to hit a moving ball with a paddle).

We then train a baseline network for both games using DQN until convergence (50 million frames). To obtain the average (mean) performance of the agent, we perform three trials of training for each game with different setting of random start (the maximum number of no-ops to be performed by the agent at the start of an episode) at $\{20, 30, 40\}$ (30 was the original setting in [Mnih *et al.*, 2015], all other parameters were set the same as the original DQN). We call the two obtained baseline agents *B-base* and *P-base*.

We perform our TL experiments in both directions: 1) Breakout transfer from Pong (*BfP*), and 2) Pong transfer from Breakout (*PfB*). In particular, we take the *fine-tuning* approach where all layers learn [Yosinski *et al.*, 2014]. For BfP we copy the weights of the first k ($k \in \{1, 2, 3\}$) convolutional layers from P-base to the corresponding layers of a new non-trained DQN agent. We initialize the other layers' weights randomly (as done in the original DQN algorithm) and then train the agent to play Breakout from scratch us-

[1] https://github.com/asrivat1/DeepLearningVideoGames



(a) BfP-k: Breakout transfer from Pong

(b) PfB-k: Pong transfer from Breakout

Figure 2: Performance of baseline and transfer agents in the Atari domain. The lines represent average reward obtained and the outlines show the standard deviation over three trials.

Table 1: TL evaluation metrics of transferring k convolutional layers in Breakout and Pong.

Agent	B-base	BfP-1	BfP-2	BfP-3	P-base	PfB-1	PfB-2	PfB-3
Jumpstart	0	0.31	-0.69	0.38	0	-0.02	0.13	2.23
Final performance	194.26	233.14	247.81	254.6	15.78	18.2	18.28	18.13
Total reward	2717.5	4074.5	4185.8	4565.4	9.16 [2]	185.91	201.55	188.93
Transfer ratio	-	149.93%	154.03%	168.%	-	2029.6%	2200.3%	2062.6%

ing the same DQN algorithm (and vice versa for PfB). Due to computational constraints, instead of training until convergence, we perform training for 30 epochs on transfer agents. A training epoch is 250,000 frames (equivalent to 7.5 million frames in total) and for each epoch, we evaluate the learning with one testing epoch consisting of 125,000 frames and record the best average test reward. Note that although the transfer training does not converge, the objective of TL is to observe initial learning improvement. We do not consider the absolute performance; a shorter training period can still give us a good description of transfer effects. Similarly, we also perform three trials (random start at $\{20, 30, 40\}$) to get the average performance for each layer transfer and obtain six transfer agents:

- *BfP-k*: Breakout agent using P-base's first k layers' weights, where $k \in \{1, 2, 3\}$ (e.g., BfP-2 refers to the Breakout agent transferring the first and second layer's weights from the Pong agent).
- *PfB-k*: Pong agent using B-base's first k layers' weights, where $k \in \{1, 2, 3\}$ (e.g., PfB-2 refers to the Pong agent transferring the first and second layer's weights from the Breakout agent).

Figure 2 shows the average performance of the transfer agent compared to the baseline agent within corresponding epochs. All transfer agents obtain better overall performance than baseline agents. Evaluation details in Table 1 are based on the four metrics where jumpstart is the difference between the first test reward of the transfer agent and the baseline agent, final performance is the mean of the last five test rewards, total reward is the sum of all test rewards [2], and transfer ratio is the ratio of total reward accumulated by the transfer agent and the baseline agent. In particular, BfP-3 has a relatively better result compared to the other two BfP agents, while for PfB agents, PfB-3 obtains the highest jumpstart but PfB-2 has the best result for the other three metrics. Except for two observed negative jumpstarts (BfP-2 and PfB-1), all transfer agents show good improvements in the four metrics. This shows that it is possible to speed up the learning in DQN via far transfer, without any alterations to its original structure.

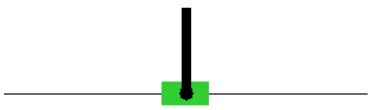


Figure 3: Visualization of the simulated Cart-pole using *pygame*.

3.2 Cart-Pole

The Cart-Pole [Michie and Chambers, 1968], in Figure 3, consists of moving a cart linearly along one dimension on a finite track. The goal is to keep a rotating pole vertically balanced by moving left or right. We use a non-linear second-order discrete physics model with a velocity-based linear fric-

[2]Note that in the game of Pong rewards are initially negative. Although this makes the improvement in the transfer ratio misleading, the results still indicate that transfer agents can learn faster.

tion model for simulating the cart-pole using the following system of equations:

$$\begin{bmatrix} \theta_{t+1} \\ \dot{\theta}_{t+1} \\ x_{c_{t+1}} \\ \dot{x}_{c_{t+1}} \\ x_{p_{t+1}} \\ y_{p_{t+1}} \end{bmatrix} = \begin{bmatrix} \arctan(\frac{x_{p_t} - x_{c_t}}{y_{p_t}}) + \dot{\theta}_t \Delta + \frac{1}{2}\ddot{\theta}_t \Delta^2 \\ \dot{\theta}_t + \ddot{\theta}_t \Delta \\ x_{c_t} + \ell \sin(\theta_{t+1}) \\ \ell \cos(\theta_{t+1}) \\ x_{c_t} + \dot{x}_{c_t} \Delta + \frac{1}{2}\ddot{x}_{c_t} {\Delta_t}^2 \\ \dot{x}_{c_t} + \ddot{x}_{c_t} \Delta \end{bmatrix}$$

$$\begin{bmatrix} \ddot{\theta}_t \\ \ddot{x}_{c_t} \end{bmatrix} = \begin{bmatrix} \frac{-mgsin(\theta_t)}{\ell} - \frac{\dot{\theta}_t \mu_p}{m} \\ a_t - \dot{x}_{c_t} \mu_c \end{bmatrix}$$

where θ_t is the pole's angle, x_{c_t} is the cart's position, and (x_{p_t}, y_{p_t}) is the location of the pole's center of mass, all at timestep t. We use constant values for the physics update rate ($\Delta = 0.01$) and gravity ($g = 9.81$). Actions a_t are in the form of positive, negative or no force where $a_t \in \{0, \pm 0.01, \pm 0.05, \pm 0.1, \pm 0.5, \pm 1, \pm 5, \pm 20\}$. For the transfer task, we change the values of the following parameters:

- distance from the cart to the pole's center of mass (ℓ)
- mass of the pole (m)
- pole's friction coefficient (μ_p)
- cart's friction coefficient (μ_c)

The system resets with the cart centered on the track and the pole's angle is drawn from a uniform random distribution in the range of $[-20°, 20°]$ where $\theta_t = 0°$ means the pole is vertically upright. If the cart hits either end of the track or if the pole angle $|\theta_t| > 30°$ the system terminates and resets. The reward (r_t) given is proportional to the deviation of the pole angle, where $r_t = (15° - |\theta_t|)/15°$. A (-1) reward is added if the cart hits the edge of the track, which has a length of 1.

Transfer is done by creating two tasks with different system dynamics. After generating the baseline performance for both tasks, we use the task with the poorer rate of convergence as the target task because it will benefit more from the transfer. The source task parameters are set to $\{\ell = 0.2, m = 0.0335, \mu_p = 0.002, \mu_c = 0.08\}$, while the target task has $\{\ell = 0.4, m = 0.134, \mu_p = 0.0005, \mu_c = 0.005\}$. Since the source and target tasks are using the same domain, this is a near-transfer problem.

With three convolutional layers, we conduct three experiments, varying the number of layers to transfer. Due to computational constraints, each experiment is trained for 250,000 timesteps only. For the cart-pole experiments, we use the notation *C-k*, where $k \in \{1, 2, 3\}$. We also have three trials per experiment. At every 5,000 timesteps, we evaluate the learned policy by computing the mean reward. Policy evaluation is done for 10,000 timesteps using greedy actions. Figure 4 shows the learning curve on the cart-pole experiments. We observe a positive transfer when transferring only the first convolutional layer, and the effects of transfer worsen as we transfer more layers. In regards to the TL metrics, each metric follows the same calculations as in the Atari domain. Table 2 shows that transferring the first layer alone has the highest final performance, total reward, and transfer ratio. Note that negative jumpstart is observed in all three experiments.

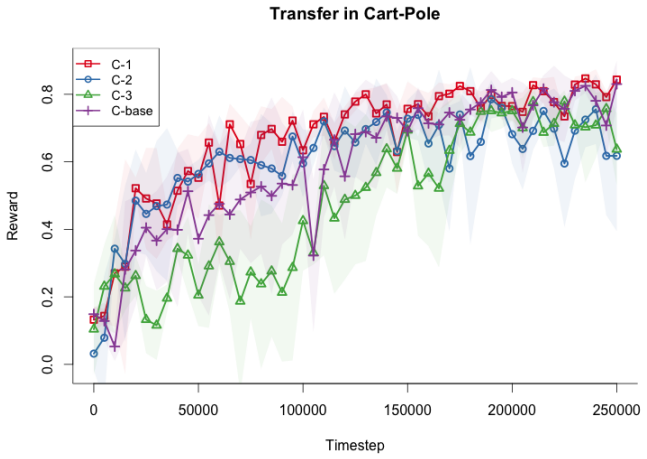


Figure 4: Performance of baseline and transfer agents in the cart-pole domain. The lines represent the mean performance and the outlines show the standard deviation over three trials.

Table 2: TL evaluation metrics of transferring k convolutional layers transfer in the cart-pole system.

Agent	C-base	C-1	C-2	C-3
Jumpstart	0	-0.0164	-0.1169	-0.0447
Final performance	0.7907	0.828	0.6817	0.7028
Total reward	30.155	33.979	30.915	24.363
Transfer Ratio	-	112.68%	102.52%	80.792%

4 Discussion

Our preliminary results open some important discussions and suggest new techniques to further study.

One phenomenon we observe in the Atari domain is in the PfB task where the transfer agents show a more consistent performance compared to BfP agents, which also happen to obtain a higher transfer ratio. We attribute this effect to the importance of source task selection — picking a better performing source task will likely provide a better transfer effect. This is supported in [Mnih *et al.*, 2015] where the DQN algorithm plays Breakout better than Pong.

In regards to the findings by [Yosinski *et al.*, 2014] on the transferability of features in deep neural networks, where higher layers learn the most generalized representation of the environment, while lower layers' representation is more task-specific — our results are inconsistent with their findings. In the Pong and Breakout transfer, the domains are different and we were expecting better results on $k = 1$ transfer, but this is not the case since lower layers like BfP-3 and PfB-2 show better performance. However, the case could also be that Breakout and Pong are more similar than we previously assumed. In the cart-pole problem we were also expecting better results in the lower layer transfer, since the source and target tasks are in the same domain, but positive transfer is only observed at $k = 1$.

Besides transferring the weights, we also look into transferring experience replay memory in the cart-pole domain, to de-

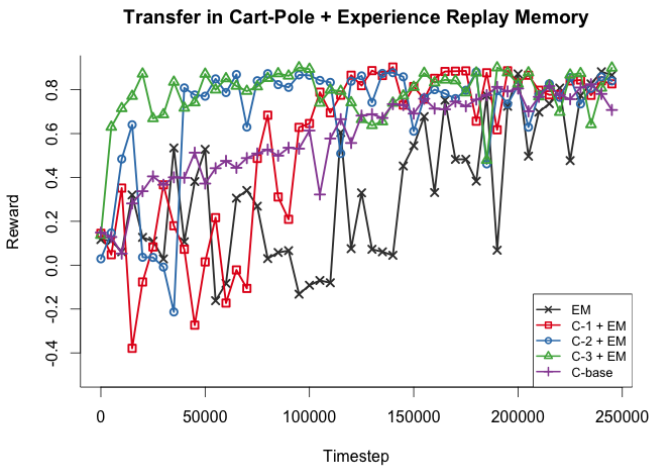


Figure 5: Transferring only the experience replay memory (EM) and transferring both weights and EM (C-k + EM) for k convolutional layers

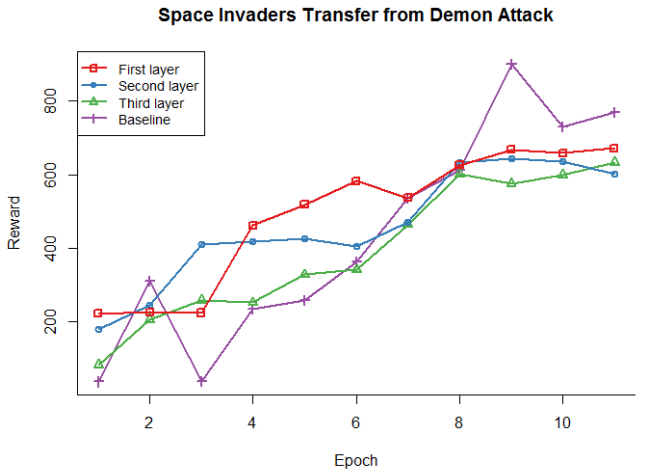


Figure 6: Game of Space Invaders transfer from Demon Attack on all three convolutional layers over a single trial. We observe a jumpstart but transfer soon hurt the performance.

termine if an agent can learn from a different data distribution, or from related experience. Figure 5 shows our preliminary results with this approach where it indicates significant positive transfer. Results also show that performance improves as we transfer experience replay memory with deeper layers, but transferring experience replay memory alone seems to have a negative effect. We are uncertain if positive transfer is present only within the same domain.

We also encounter a *negative transfer* (transfer hurt the performance) when transferring between a different pair of Atari games, specifically *Space Invaders* transferring from *Demon Attack*, given that Demon Attack has better initial performance in DQN. These two games satisfy the similarity requirement for source/task game selection (both are shooting games that the player can move in all directions, try to shoot enemies, and share six same game actions). Figure 6 shows the result for training for 11 epochs (2.75 million frames). Although we observe jumpstart, negative transfer occurs after only 6 epochs (1.5 million frames). However, since we only perform one trial per layer and the performance curve is less stable, the result can only be viewed as preliminary.

5 Conclusion and Future Work

Our initial progress in the Atari and cart-pole domains show that TL is a promising method to improve the time efficiency of the DQN algorithm. However, due to the time consuming nature of these experiments, and our limited high-computing resources, more time is needed for further research.

We would like to extend this work in four interesting directions. First, we intend to perform more trials in our current domains, and test transferability on other Atari games. We will explore transferring the experience replay memory in the Atari domain and also test if the performance will be affected if we shuffle the weight ordering. Second, we will attempt to look into the potential of enabling knowledge selection for each layer. In particular, instead of transferring all weights from one layer to another, if we can identify and transfer specific "helpful" weights from source tasks to target tasks, we should expect better transfer effects. Third, as the source/target task selection problem still remains an open question in transfer learning, finding a more robust selection mechanism could also potentially help to achieve more efficient performance. This can be tested by performing zero-shot learning on the transfer agents — how much can they learn without explicit training but only the knowledge from their baseline agent. Ideally we should expect good generalization between games if a robust selection mechanism is used. Lastly, even though we have only performed near-transfer between one simulated cart-pole model to another, results show that TL can accelerate learning between two cart-pole models with very different dynamics. This indicates that TL can also be used to transfer knowledge between a simulated and a real cart-pole system, even if the cart-pole simulation does not accurately model the real system. Future work will investigate learning in simulation and then transferring to physical robots.

Acknowledgments

The authors thank Anelia Angelova for her useful comments and suggestions, and Brandon Kallaher for technical support. This research has taken place in the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL research is support in part by grants from NSF IIS-1149917, NSF IIS-1319412, USDA 2014-67021-22174, and a Google Research Award.

References

[Abadi *et al.*, 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp,

Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Bellemare *et al.*, 2013] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 06 2013.

[Bucilu *et al.*, 2006] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541. ACM, 2006.

[Girshick, 2015] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[Liang *et al.*, 2015] Yitao Liang, Marlos C Machado, Erik Talvitie, and Michael Bowling. State of the art control of Atari games using shallow reinforcement learning. *arXiv preprint arXiv:1512.01563*, 2015.

[Michie and Chambers, 1968] Donald Michie and Roger A Chambers. BOXES: An experiment in adaptive control. *Machine Intelligence*, 2(2):137–152, 1968.

[Mnih *et al.*, 2015] 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.

[Nair *et al.*, 2015] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

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

[Razavian *et al.*, 2014] Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[Rusu *et al.*, 2015] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[Shinners, 2011] Pete Shinners. PyGame, 2011. Software available from pygame.org.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[Taylor and Stone, 2007] 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.

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

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.