Investigating Policy Adaptations of Deep Q-Learning Autonomous Driving Agents with Transfer Learning

Jonathan Xu
North Carolina School of Science and Mathematics
jonxu100@gmail.com

Abstract

Reinforcement Learning is a popular AI algorithm used to teach an agent to learn about its environment by exploring it. We investigate Deep Reinforcement Learning agents applied to autonomous driving in the Car Racing Track environment from OpenAI gym. Reinforcement learning is very costly–training agents from scratch in a complex state space consumes large amounts of time. Transfer learning seeks to remedy this drawback by adapting knowledge from agents that have been trained in similar environments. In this paper, we investigate the effects of direct transfer learning with a Deep Q-Learning agent pretrained to solve the Car Racing Track environment. The agent begins training in the altered environments with weights from its original policy. Our altered environments are generated by adjusting parameters of the original environment such as the Track Width, Friction, and Coloring of the track and background. To benchmark the effect of transfer learning in these altered environments, we simultaneously train a null agent from scratch and compare the policies after every episode of training. We analyze the resulting reward graphs to understand the effects of transfer learning on learning rate and policy adaptation.

Introduction

Deep Reinforcement Learning (RL) is a state of the art method to train AI agents learning in an environment. Every environment includes some reward to indicate the success of an agent, while the agent develops a policy to maximize these rewards. By exploring extensively in its environment, an agent gains knowledge about the environment via reinforcement learning. However, in environments with state inputs such as raw images, an agent cannot make sense of its environment very well.

Deep Learning models enable the agent to represent these image states as numerical knowledge, which can then be fed into reinforcement learning algorithms. As such, Deep Reinforcement Learning has direct applications to computer vision, particularly in robotics [Kober, Bagnell, and Peters2013], solving video games [Mnih et al.2013], and automated vehicles [You et al.2017, Sharma et al.2019]. We will examine an application of Deep Reinforcement Learning in an approximated environment of autonomous driving.

Deep Reinforcement Learning is very expensive, requiring large amounts of time to train, especially in large and complex environments such as the video game DOTA 2 [OpenAI et al.2019]. Simulations of real world environments are no exception, being arguably even more complex than a video game. With reinforcement learning, an agent can only learn to map a state and action by experiencing that mapping explicitly. Thus, as the state space of the environment increases, it becomes very time consuming for an agent to experience enough states and actions to build its knowledge. One game with a very large state space is chess, which was mastered by reinforcement learning recently with the AlphaGo and AlphaZero project [Silver et al.2017]. Many environments also feature very sparse rewards, which lengthens training time because it is significantly more difficult to find successful policies in these environments.

Recently there has been a lot of work investigating deep reinforcement learning agents and their ability adaptability to new environments more quickly. Some current approaches include transfer learning [Sharma et al.2019] and Sim2Real [Sadeghi et al.2018]. Sim2Real is a emerging field focused specifically on applying transfer learning from virtual environments to real world environments. Despite these prior attempts, it is still not well understood where transfer learning will be successful or unsuccessful.

In this paper, we examine the application of transfer learning with Deep Reinforcement Learning agents to self driving vehicles in a 2D virtual racing track environment. We use the 2D Car Racing track simulator from the OpenAI gym as the environment for our agent. We apply an open source pretrained Deep-Q Reinforcement Learning agent as the pretrained agent to test transfer learning, with an untrained "null agent" serving as a baseline. We altered the coloring of the environment, manipulated the physics, and varied the dimensions of the track to examine how the agent would adapt to these changes relative to the null agent. It's important to use a null agent as a control in order to gauge how effective transfer learning is in these environments. While our environment much simpler real world, the results of our ex-

periments are likely applicable to agents solving real world autonomous driving environments.

Related Work

Video games are an appropriate place to test Deep Reinforcement Learning agents, as every game involves some sort of reward that an agent can maximize. The success of efficient image processing with deep neural networks has led researchers to investigate combining Deep Learning models with reinforcement learning. In 2013, by combining a convolutional neural network with a Q-learning algorithm, researchers were able to solve Atari 2600 games to a level beyond human experts [Mnih et al.2013]. More recently, Deep Reinforcement Learning has been applied to solve real world environments, such as flight simulations [Kang et al.2019] and robotics [Kulhánek et al.2019]. Despite its successes, the policy of a Deep Reinforcement Learning agent is very sensitive to small alterations in its environment [Müller et al.2018]. Transfer learning studies how well knowledge from one policy will more efficiently or robustly train to adapt to a similar environment. In this paper, we will investigate the effects of transfer learning with a pretrained Deep Q-Learning agent in a game-like emulation of a race track.

Transfer learning in reinforcement learning is a well-studied area [Zhuang et al.2020]. There are a few general approaches to this problem such as imitation learning [Pan et al.2020], GAN [Gamrian and Goldberg2018, Zhang et al.2018], and direct transfer learning [Müller et al.2018]. In *Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation* [Gamrian and Goldberg2018], the authors apply a generative adversarial network (GAN) to remap images between different levels of Atari games, allowing an agent to transfer its knowledge from the first level to future levels. They found that significant capabilities from previous training could be carried over to different levels, demonstrating transfer learning's effectiveness in similar environments. In our Car Racing Track environment, we utilize a direct approach to transfer learning as opposed to using a GAN network. We generate alternate test environments by selectively changing parameters and observing how a pretrained agent responds and learns a successful policy compared to a null agent.

Approach

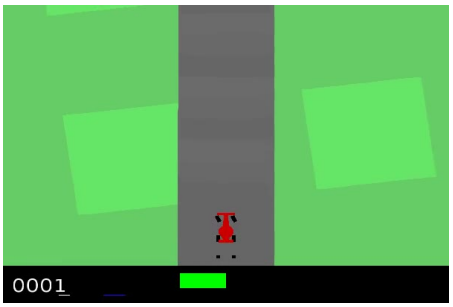


Figure 1: An image of the original environment.

In this section we describe our transfer learning approach in a racecar environment. The first step was to select a suitable environment–the OpenAI gym Car Racing track (Shown in Figure 1). The environment's parameters are easily adjusted in order to investigate transfer learning of an agent to our customized altered environments. We then identified a Deep Q-Learning agent trained in the car racing track environment, which we used as our pretrained agent to test transfer learning. Finally, we chose several parameters to conduct several experiments with, investigating the effectiveness of the transfer of knowledge to solve the new environments.

Original Environment

The original environment that is solved by our pretrained agent is the OpenAI gym Car Racing Track environment. The agent in this environment controls a simple car with 4 possible actions: do nothing, accelerate left, accelerate right, and accelerate forward. To evaluate an agent, the environment uses a reward function.

The reward function of this environment is simple. Each track tile visited awards $+1000/N$ where N is the total number of tiles with a discount factor of -0.1 every frame of the simulation. The reward is effective in measuring an agent's success for our purposes. It heavily rewards completeness of the track while encouraging speed by punishing the agent the longer it takes.

We can represent the environment as a Markov Decision Process. To traverse the track, an agent utilizes reinforcement learning to solve the Markov Decision Process. A Markov Decision Process is based on the idea that the future states are dependent on the current state. With a set of possible actions in the current state, there are only a finite number of future states the agent can directly reach. A Markov Decision Process is made up of several components: the states, the rewards of each states, and actions to traverse states. An additional factor may represent how significantly rewards from future states several steps into the future are weighted in the current state.

As mentioned before, the reward function of this environment is $+1000/N$ where N is the total number of tiles with a discount factor of -0.1 every frame, and the actions are represented by the commands do nothing, accelerate left, accelerate right, and accelerate forward that can be given by the agent. The states in the car racing environment are the images of the vehicle and its surroundings. Note that normally some vector or numerical representation can be given to each state. In this case the agent only receives raw images, meaning a deep learning model can be utilized to model each frame as a state. Given these parts of the Markov Decision Process, an agent is able to use reinforcement learning to identify favorable states and actions to take to maximize the reward from the environment.

Our Reinforcement Learning works by updating the weights within the agent's policy after every episode of training. Every episode is comprised of several rollouts, which means one iteration of the simulation. A rollout is comprised of several steps, which recall from the Markov Decision Process are the state-actions the agent undergoes.

The following are parameters that we will vary in the experiments.

- **Road Width:** The pixel width of the road for the agent to traverse. Default=60.
- **Friction Value:** The friction value for the road. Default=1.0.
- **Background Color:** RGB value for the color of the background. Default=[0.4,0.8,0.4] (Green).
- **Track Color:** RGB value for the color of the track. Default=[0.4, 0.4, 0.4] (Grey).

Model

To solve the original OpenAI *Car Racing Track* environment described in the previous section, we utilize a Deep Q-learning agent. An agent with a policy that already solves the environment will serve as the pretrained agent in each of the transfer learning experiments. We used an open sourced Deep-Q-Network [pekaalto] that is trained to nearly perfectly traverse the Car Racing environment. It's learning rate is 0.0004 and utilizes a ADAM optimizer. The structure of the model is shown in Fig. 2.

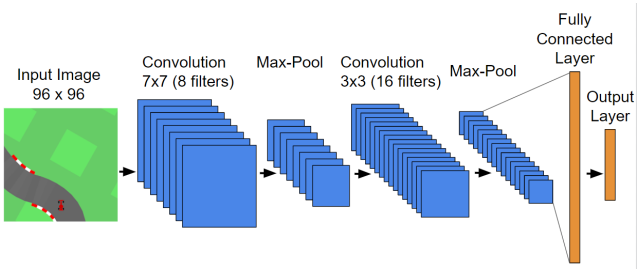


Figure 2: The convolutional neural network visualized in a diagram.

The model uses a convolutional neural network in order to process the input image to determine the correct next action for the agent to take. While the simulation displayed by the OpenAI gym engine is 600 x 400 pixels (the track width is adjusted on this scale), the agent is limited to a 96 x 96 pixel image input. The network extracts features such as the border between the track and background from these input images by utilizing convolutional layers with ADAM optimization. The following layers assess the results of the convolutional layers to calculate the perceived optimal action. Overall, the model uses a learning rate of 0.0004 to train.

Altered Environments

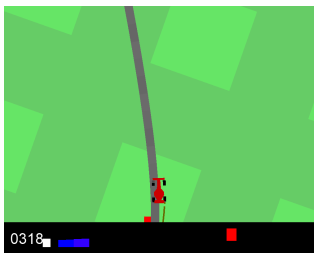
The environment has a set of several parameters that we altered to test the effectiveness of transfer learning. In most cases, we only alter one parameter in each experiment in order to isolate the results, but occasionally we alter multiple parameters to investigate if the results still remain consistent.

- **Original Environment:** Width=60 pixels, Friction=1.0, Road Color=[0.4, 0.4, 0.4] (Grey), Background Color=[0.4,0.8,0.4] (Green)
- **Small Track:** Dropped the width of the track to 5 pixels, which is notably thinner than the agent. The current policy of the pretrained agent is to follow alongside the border of the track. By drastically reducing the track width to be smaller than the width of the car, we test if the agent's original policy is still beneficial to learning.
- **Wide Track:** Increased the width to 400 pixels. Again, the current policy relies on identifying some sort of border. With such a large width, the agent will be unable to determine a border as the track essentially forms a circle. Such a large difference in the nature of the track and a target policy puts transfer learning to the test.
- **Slippery Track:** Significantly reduced the friction of the environment from 1.0 to 0.1. Here, we mimic an "icy" environment. Altering the physics of the environment interferes with the agent's mobility given its current policy. Here we're isolating the agent's ability to adapt its control of the vehicle as opposed to detecting the road.
- **Small Slippery Track:** Combined changes in friction with changes in track width. In each of the previous environment variations, we isolated the differences in the track detection and maneuvering of the vehicle. Altering both simultaneously is also of interest to investigate how quickly and effectively transfer learning is able to accelerate learning policies to deal with heavily altered environments.
- **Altered Color Environments:** The static coloring of the background and track were altered to selected colors. We then experimented with randomized track or background color between rollouts.
  - **Purple Background:** Altered the background or "grass" color of the environment to [0.8, 0.0, 0.8] (Purple). The green value of the background is likely of high importance to the agent's policy. By altering the background to the opposite color of green, purple, we test how well the agent is able to leverage its current policy to learn a policy for a drastically different environment.
  - **White Background:** Altered the background or "grass" color of the environment to [1.0, 1.0, 1.0] (White). In contrast to the last experiment, we choose a color that is relatively similar to green, white. The pretrained agent is theoretically still able to utilize its policy on the green rgb value of the background, so this environment heavily examines the transfer of knowledge.
  - **Yellow Track:** Altered the track color of the environment to [0.8, 0.8, 0.0] (Yellow). Here, we're trying to confuse the pretrained agent again. Yellow is a very close color to the original background, green, so it should be quite difficult for the agent to differentiate the border. The ideal policy here should clearly focus on the red rgb value, whereas the original policy of the agent almost certainly doesn't.
  - **Blue Track:** Altered the track color of the environment to [0.0, 0.0, 0.8] (Blue). We now change the track to blue for similar reasons we changed the background to

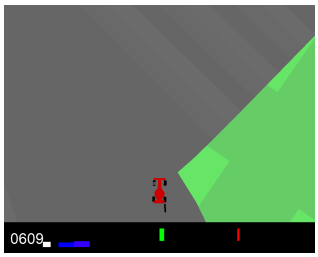
white. The difference between the original track and background rgb was in the green value, and that holds in this experiment. The transfer of knowledge from the pretrained agent should be more straightforward here.

- **Random Background:** Changed the background color to a random color every rollout. Theoretically, it should be possible for the agent to identify the track based purely on the pixels of the track, regardless of the border. The purpose of this experiment is to identify whether or not the agent is able to create a policy that allows for this, and whether or not the pretrained agent is more effective in doing so.
- **Random Track:** Just as with the previous experiment, we attempt to motivate the agent into learning a policy that focuses purely on the background color to identify its location. The transfer of knowledge is interesting in these cases, as the pretrained agent should already have some sense of identifying the road and background regardless of the other.

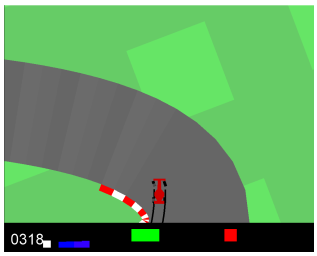
The resulting environment from some of these alterations are shown in Figure 3 below.



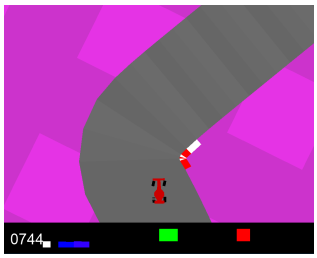
(a) The Small Track environment



(b) The Wide Track environment



(c) Although not visible from this image, this is the Slippery Environment.



(d) Purple Background Environment. Several other orientations of colors were tested as well.

Figure 3: Images of altered environments.

Evaluation

Our goal is to investigate the effectiveness of transfer learning from the original environment into altered environments. The original environment consists of a track of width=60 and friction=1.0, with the rgb values of the background and track as [0.4, 0.8, 0.4] and [0.4, 0.4, 0.4] respectively (rgb values given on a scale with max 1.0). In each experiment, the null agent is an agent that has no prior training, while the pretrained agent had been trained for about 150,000 steps in the original environment. Each agent had a learning rate of 0.0004 and utilized the adam optimizer. We train the null agent and pretrained agent in each altered environment, evaluating each agent respectively after every episode of training. Typically an experiment consisted of 40 episodes of training, although in some cases experiments lasted up to 60 episodes. Evaluating each experiment is done by running 20 rollouts on every episode and averaging the rewards across the rollouts. By doing so, we are able to track through each episode how quickly and how well an agent is able to adapt to its environment. Comparing the null agent's results and pretrained agent's results gives us insight as to how effective transfer learning was for learning a proficient policy in an altered environment.

The current policy of the pretrained agent is simple: find the left side of the road and maintain a distance from the right side. The agent is able to smoothly navigate the environment with the exception of sharp turns, where it tends to cut corners and even occasionally spin off. The agent's ability to navigate the environment is heavily dependent on the agent's ability to identify the bounds of the road. As such, altering the dimensions of the road in certain experiments is an interesting prospect to investigate transfer learning. The pretrained agent must also heavily leverage the static coloring of the road and the "grass" surrounding it. Thus, in some experiments we alter the color of the road and the background individually. Inverting the color of the road and the background was of special interest, as it seems taking the opposite colorings contradicts the learned policies of the pretrained agent as much as possible. Additionally, we altered the friction of the road as a simulation of "icy" roads. While the agent should be able to identify the bounds of the roads, we evaluate how well the agent is able to alter its policy to refine its motor control.

There are a couple signs to determine if the transferred knowledge is helpful. The pretrained agent could receive a "jumpstart", meaning at episode 0 it already performed better than the null agent. This meant that the transferred knowledge was immediately relevant to the altered environment. The transferred knowledge could also be considered useful if the ending average reward would be higher than that of the null agent. A higher final reward means the transferred knowledge gave the pretrained agent an advantage in training compared the random noise of the null agent. If the pretrained agent's learning speed matches that of the null agent, we can consider the pretrained agent's knowledge as not useful. In the case of a pretrained agent consistenly performing worse than the null agent by the end of the final episode, we can conclude the transferred knowledge to be detrimental.

Results

We now present the results of each experiment with graphs comparing the rewards of the pretrained agent and the null agent vs the number of episodes trained. From the OpenAI gym documentation, the Car Racing Track environment (and these variants) are considered "solved" when an agent can achieve a score of 900 consistently. As a baseline in the original environment the pretrained agent achieves a score

of 800-900 consistently.In every graph, the x-axis represents the number of episodes of training and the y-axis represents the average reward for a rollout over 20 trials. With the data, we are able to deduce the effectiveness of transfer learning for each of the altered parameters by following the guidelines outlined in the evaluation section.

Track Width

In the first experiment we test the **Small Track** environment, where the track is narrower than the car itself.

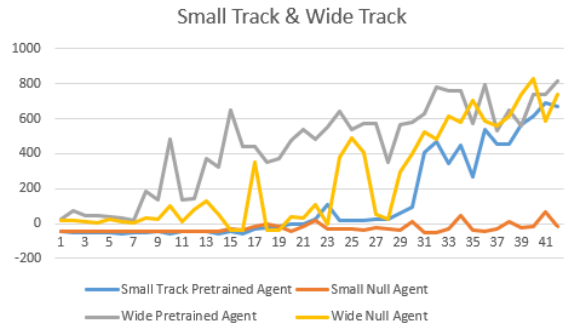


Figure 4: The rewards of the pretrained agent and null agent in the Small Track and Wide Track environment

As shown in Fig. 4, the pretrained agent exhibits a clear learning advantage over the null agent. The knowledge of the background is likely leveraged by the pretrained agent, which only needs to adjust its policy to recognize the thinner track above and below it. The pretrained agent is given no jumpstart, which is not surprising because the original policy of following the track border is not applicable to such a narrow track. Nevertheless, the knowledge of the original environment helps the training of the agent rapidly, whereas the null agent cannot learn very effectively by randomly exploring such a delicate environment.

In the second experiment we investigate the other extreme with the **Wide Track** environment–a track wide enough that it takes up nearly the entire screen.

With a track that covers virtually the whole screen, the pretrained agent is initially able to learn to adapt to this environment more quickly as Figure 4 suggests. This is likely due to the the pretrained agent's knowledge of the original environment, providing it with a jumpstart to its learning.

The null agent struggles for quite a few episodes to make any progress, but eventually learns very quickly to traverse the environment just as proficiently as the pretrained agent. Because the track is so large, the null agent was able to stay on the track longer and thus undergo more steps every episode. The Wide Track environment in this way rapidly accelerates the null agent's learning compared to null agents in other environments as we will show later. After about 30 training episodes, the two agents plateau into receiving about the same average reward.

Recall that a strong score in these environments is about 800-900. Clearly both the null and pretrained agent plateau to scores between 600-700 consistently, indicating their policies were far from optimal. The agents simply learned to roam about randomly within the confines of the road, which were very lenient and required little skill to traverse. Thus, there was no need for the agents to systematically visit every tile to yield a relatively large reward. We can consider this a local maxima–it is very easy way to gain a fairly large score with a very simple policy.

Varying the track width to two extrema gave two contrasting results in transfer learning. The ideal policy for the Small Track is very similar to the Wide Track–simply follow the road and adjust at turns as needed. The transfer learning yielded very successful results in the narrow track case, where the agent was forced to approach the track slowly to maintain precision over the track. With a wide track, the results were less successful. The goal policy for a track that covers the screen initially would be to find the border and follow it. This is much more complex and while the pretrained agent may have a notion of traveling on the road, it ultimately could not improve much from its local maxima.

Friction

For the first experiment in this section we set the track width to the default 60, and instead drastically alter the friction. We expect the pretrained agent to perform significantly better because it already has some understanding of the bounds of the track. However, the reduction of friction will likely cause the agent to struggle at turning points, especially sharp turns.

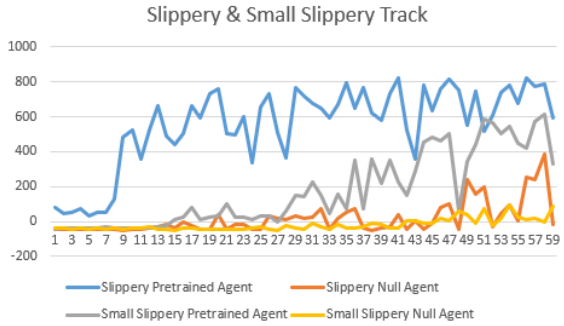


Figure 5: The rewards of the pretrained agent and null agent in the Slippery Track and Small Slippery Track environment environment

Figure 5 shows the results are exactly as we expect. The pretrained agent has a minor jumpstart to the Slippery Track environment, then rapidly learning to achieve consistently high results after about 15 episodes of training. The null agent struggles to learn on this track, only adapting slightly near the end of the training. This experiment clearly demonstrates the effectiveness of transfer learning. The agent's ability to adapt its model very quickly gives it a significant advantage over an agent learning to solve the environment from scratch.

In the previous experiment, setting the width to the default value gave the pretrained agent a significant advantage. We now alter the track with to 5 to create a small slippery track. We combine these two alterations to the environment

to challenge the pretrained agent's ability to use its prior knowledge. In such a difficult and volatile environment, we expect the null agent to struggle to learn a policy. We anticipate the pretrained agent learns will learn in a similar fashion to the Small Track, although slower.

The pretrained agent begins with no jumpstart just as in the Small Track environment (Figure 4), likely because the environment is drastically different and volatile. Surprisingly, the pretrained agent begins learning much earlier in this environment than on the small track, although this could be by random chance from exploration. Overall, the result in the Slippery Small Track environment is analogous to the Small Track environment. The pretrained agent's final average of about 500 in the Slippery variant is slightly less than the regular Small Track environment, in which the agent yielded an average of about 600 at the end of training. The null agent struggles in both environments, demonstrating the effectiveness of transfer learning when altering these parameters.

Background Color

Here we will examine the effects of altering the background color while maintaining the original track width and friction values. We first tested the White Background environment. We theorize the green RGB value of the background color is heavily weighted in convolutional neural network. As such, an agent in the White Background environment should still be able to take advantage of the difference in the between the green RGB values of the background and the road. It is fair to expect the pretrained agent to perform very well while the null agent will have to learn from scratch. The

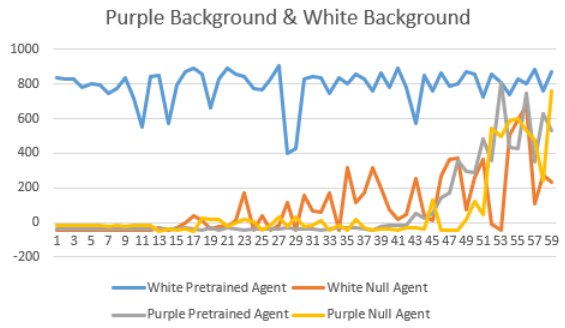


Figure 6: The rewards of the pretrained agent and null agent in the White Background and Purple Background environment

results in Figure 6 indicate exactly the phenomenon. The pretrained agent exhibits a significant jumpstart compared to the null agent, showing that its policy was well adaptable to the white background. The initial average rewards around 800 in the White Background environment are even on par with the pretrained agent "solving" the original environment, which recall was an average reward of nearly 900. Additionally, the pretrained agent becomes more consistent in reaching high rewards with more and more training, indicating that it's further optimizing its policy.

The reason for such a massive jumpstart is likely as we hypothesized. With a white background, the green RGB value is very similar between the original and altered environment. This means the agent is able to use its policy exactly as before and achieve strong results. The null agent was much less successful, slowly learning from scratch a policy in the environment. This is essentially the equivalent of training the null agent in the original environment. The transfer learning was very successful in this experiment due to the similarities in the environments.

Next we test the pretrained agent's ability to learn with a background the inverse color of green–purple.

The results in Figure 6 support the idea that the agent was able to use its policy in the white background experiment due to the green value being similar. With a green RGB value of 0 in the purple background color, the agent essentially has lost all bearings of its surroundings. Its learning rate is roughly the same as the null agent, only achieving good results after 55 episodes of training. The agent has to virtually relearn its entire policy making it equivalent to a null agent that learns from scratch, making transfer learning not ideal in this scenario.

Road Color

We've examined the effects of altering the background color, so it is interesting to see if changing the road color instead while keeping the original track width and friction values will yield similar results. First we consider the target environment Blue Track. Recall with the white background experiment we emphasized the green RGB value of these environments, whereas in this experiment we focus on the distinct blue RGB values. The road and the background clearly have significantly different blue RGB values in both the original and blue track. As such, we expect this experiment to be very similar to the white background experiment, an immediately successful pretrained agent while the null agent struggles.

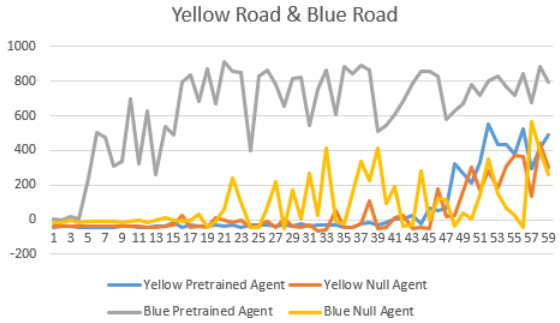


Figure 7: The rewards of the pretrained agent and null agent in the Blue Track and Yellow Track environments

The pretrained agent surprisingly starts without a jumpstart but immediately makes up for it by learning a proficient policy by the 17th episode in the Blue Track environment. Our hypothesis that the pretrained agent would immediately in the altered environment was false, but we were correct

to anticipate the pretrained agent could very quickly and effectively adapt to the blue track. From the previous experiments, we know the agent is able to track the background in addition to the road. Thus, the agent only needs to slightly alter its CNN layers until averaging a score of 800, indicating a strong mastery of the environment.

The comparison to the null agent demonstrates just how effective the transfer learning is in this case. The null agent struggles to receive any reward, and eventually improves only slightly reaching a peak reward of only nearly 600 in one episode. The end policy of the null agent is far less successful than that of the pretrained agent.

A blue road, however as discussed before, created an environment where the target policy is very similar to the original policy. A more challenging environment where the road's color is very similar to the background color could make transfer learning much less effective. To test this hypothesis, we ran an experiment with a yellow road, again shown in Figure 7.

The results are as we hypothesized. The pretrained agent's knowledge is not useful in this environment because it is much more difficult to determine the border between the road and the background. In fact, the null and pretrained agent behave almost identically, illustrating that the knowledge from the pretrained agent's model is not useful to adapt to this environment. By investigating the Yellow Track and Purple Background environments, we can conclude that in environments with drastically different parameters, the pretrained agent's knowledge of the original environment is not useful in the new environment.

Randomized Environments

With such contrasting results between different colorings of the track and background, we employ an environment with randomized colors between rollouts to determine if the pretrained agent is able to adapt its model to a variety of colors. The goal is for the agent to be able to learn a policy that no longer focuses on the background color. Instead, the agent would ideally only need to observe the road immediately around it. We expect the pretrained agent to use its original policy to rapidly develop this generalized policy.

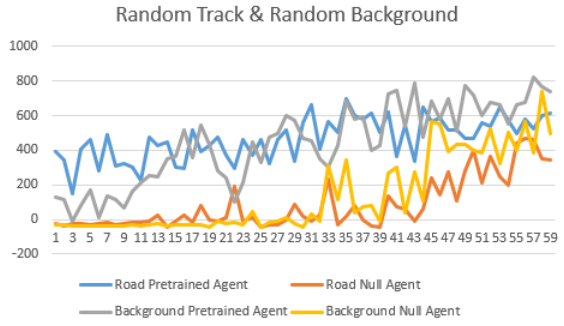


Figure 8: The rewards of the pretrained agent and null agent in the Random Background and Random Track color environment

Figure 8 exhibits interesting results. In general, the pretrained agent was on average fairly effective across all episodes, improving at a consistent pace over episodes to reach a final average of about 800. This experiment demonstrated clearly the effectiveness of transfer learning. The pretrained agent was given a jumpstart and an initial surge in its rewards improvement rate. The null agent started very poorly, but eventually was able to rapidly learn its own policy and nearly catch up to the pretrained agent.

We also use the Random Track environment to observe whether or not the agent can learn to maneuver by focusing on the background.

The pretrained agent is given a jumpstart from its starting at a relatively solid score of about 300 and slowly learning to average nearly 600. The null agent struggles at the onset of training, but soon rapidly learns to almost match the pretrained agent. Prebiously, the pretrained agent's policy likely heavily weighted the road to detect the agent's location while only using the background as an edge detector. If this is the case, it is understandable that the pretrained agent was unable to quickly learn a policy that uses the background color to determine its state.

Conclusions

In this paper we investigated the effects of transfer learning of a Deep Q-Learning agent pretrained in the car racing environment from OpenAI gym. The Car Racing Track environment from OpenAI is a simple yet reasonably realistic emulation of the real world to test autonomous driving agents. We altered the environment by varying the track size, the friction value, and the color scheme of the track and road. The environments we chose to test were extreme to test the boundaries of transfer learning. We found transfer learning to be extremely effective in environments that did not undergo significant transformations. Altering physical parameters such as the track width and friction lended well to transfer learning, as the pretrained agent's policy for edge detection and maneuvering needed only small modifications. The pretrained agent struggled, however, in the Purple Background and Yellow Track because its original policy was not applicable to the inverted colorings. Transfer learning was unsuccessful in these environments–the pretrained agent exhibited no advantage over the null agent in training.

Acknowledgements

I would like to express my deep gratitude to Dr. Matthew Guzdial, my research supervisor, for his patient guidance, enthusiastic encouragement and constructive feedback on this research work. My accomplishment of this project would not be possible without his continuous support. I offer my appreciation for his willingness to give his time so generously.

I would also like to thank OpenAI. Their open source gym environments served as an accessible resource and inspiration for this work.

References

Gamrian, S., and Goldberg, Y. 2018. Transfer learning for related reinforcement learning tasks via image-to-image translation. *CoRR* abs/1806.07377.

Kang, K.; Belkhale, S.; Kahn, G.; Abbeel, P.; and Levine, S. 2019. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 International Conference on Robotics and Automation (ICRA)*, 6008–6014.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32(11):1238–1274.

Kulhánek, J.; Derner, E.; de Bruin, T.; and Babuška, R. 2019. Vision-based navigation using deep reinforcement learning. In *2019 European Conference on Mobile Robots (ECMR)*, 1–8.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing atari with deep reinforcement learning. *CoRR* abs/1312.5602.

Müller, M.; Dosovitskiy, A.; Ghanem, B.; and Koltun, V. 2018. Driving policy transfer via modularity and abstraction.

OpenAI; :; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; Józefowicz, R.; Gray, S.; Olsson, C.; Pachocki, J.; Petrov, M.; de Oliveira Pinto, H. P.; Raiman, J.; Salimans, T.; Schlatter, J.; Schneider, J.; Sidor, S.; Sutskever, I.; Tang, J.; Wolski, F.; and Zhang, S. 2019. Dota 2 with large scale deep reinforcement learning.

Pan, Y.; Cheng, C.-A.; Saigol, K.; Lee, K.; Yan, X.; Theodorou, E. A.; and Boots, B. 2020. Imitation learning for agile autonomous driving. *The International Journal of Robotics Research* 39(2-3):286–302.

pekaalto.

Sadeghi, F.; Toshev, A.; Jang, E.; and Levine, S. 2018. Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4691–4699.

Sharma, S.; Ball, J. E.; Tang, B.; Carruth, D. W.; Doude, M.; and Islam, M. A. 2019. Semantic segmentation with transfer learning for off-road autonomous driving. *Sensors* 19(11):2577.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR* abs/1712.01815.

You, Y.; Pan, X.; Wang, Z.; and Lu, C. 2017. Virtual to real reinforcement learning for autonomous driving. *CoRR* abs/1704.03952.

Zhang, M.; Zhang, Y.; Zhang, L.; Liu, C.; and Khurshid, S. 2018. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 132–142.

Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; and He, Q. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE* 1–34.