Comparing Transfer Learning Methods: Deep Q Networks vs NEAT

by

Sam Dunaway

In Partial Fulfillment of the Requirements

for the Degree of Bachelor of Computer Science

Rose-Hulman Institute of Technology

Terre Haute, Indiana

February 28, 2021

Abstract:

Transfer learning is a way for models to learn to complete a task and store a knowledge base in order to apply that knowledge to a different, but usually related problem. Transfer learning can enable companies and researchers to use artificial intelligence where it may normally be prohibitively expensive, as gathering enough data to train the networks worth it can be prohibitively expensive. Transfer learning addresses this by learning an easier problem, and using that knowledge to learn a different, often more difficult problem. Transfer learning can be done with many different methods, and used with many types of problems. This paper compares how well Deep Q Networks (DQN) and networks trained by NeuroEvolution of Augmenting Topologies (NEAT) transfer from a basic car racing game to a more difficult version, comparing both the initial transfer and the transfer after having trained on the more difficult version. Since the approaches are intrinsically different, as DQN is gradient descent and NEAT is population based, I will discuss the experimental designs and my approach to provide fair comparisons to avoid misleading results. While not getting any statistically significant results, trends in the data show that NEAT performs better when little or no training is done on the target task.

1. Personal Motivation:

The reason I chose this topic is because I thought the idea of comparing two completely different methods was interesting. The population based approach was compelling to me because the biological evolution comparisons are not typical of technical problems. Also, the diversity that NEAT encourages through the population approach is an interesting topic and one that I wanted to explore more. Gradient descent is one of the most common methods, so it made a natural comparison. I have prior experience with Q learning, and DQNs are a natural extension of the idea.

2. Background:

There are many ways to perform machine learning with artificial neural networks, but the one I have chosen for this paper is reinforcement learning. Since there isn't much work comparing methods of transfer, I wanted to choose methods to compare transfer that would emulate the broader field, so I chose gradient descent, a very common approach, and population based, which has risen in prominence recently. OpenAI Gym makes this task of comparing these two approaches easier by having lots of environments to choose from, which I ended up choosing a continuous action space and high dimensional state space environment of Car Racing v0.

Artificial intelligence methods for training artificial neural networks (ANNs) generally fall into three categories, supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is excellent at taking labeled data to "train" a model, then using that to classify new data. Unsupervised learning, which trains on unlabeled data, classifies the data by observing the structure or distribution of existing data to unearth potentially unseen patterns or connections. An example of unsupervised learning would be clustering natural groups in data. Reinforcement learning, in contrast, is training agents in an environment on how to take actions to maximize reward. Each of these methods has applications in specific circumstances. Reinforcement learning can be differentiated from supervised learning because the training data does not need to be labeled, and likewise different from unsupervised learning because unsupervised learning does not train based on outputs of any reward function. The focus of this paper is on two different types of approaches that exemplify reinforcement learning, with NEAT using generations as its revision rather than episodes, like conventional reinforcement learning does. The investigation is regarding how transfer learning will perform when using them. Transfer learning is a machine learning method of using a model to solve a task and building a base of information about the task and how to do it, or do its components, then using that information to learn a different task. Transfer learning is a field with huge potential for growth, as the growing popularity of artificial intelligence is

attractive to many companies, but using conventional strategies may not be attainable. Companies in the medical field may have trouble acquiring data of patients due to painful appointments, or companies selling shoes to vegan women may have very little data to work with, or gathering data for AC130 flight durations could be prohibitively expensive. Transfer learning can be a solution, as one of the primary goals is to make artificial intelligence cheaper and easier.

An unexplored field in transfer learning is comparing different methods of transfer learning to different types of problems. To approach this, I wanted to emulate a situation of transfer that may be closer to real life scenarios, to get maximum usefulness from the results. This led me to choose a problem with a continuous action space and a high-dimensional state space. My interest in video games made CarRacingV0 from OpenAI Gym an excellent choice.

OpenAI Gym [15] is a set of environments that are used by many papers in the artificial intelligence field to test models. It has many different types of problems, as well as many environments for each type of problem. Car Racing V0 is one of these, known as a simple continuous action space environment. The maximum possible score is 1000, and it is a top down perspective driving a car around a simple track. For agents, the commonly accepted score for solving Car Racing V0 is 900. It is also easy to customize environments in OpenAI Gym, so this makes it an appealing choice for transfer learning, as the problems can be modified to suit transfer.

My choices for the methods stem from my interest in finding a niche method that isn't as popular as other methods. Gradient descent is very widely used and often applied to many problems without thought, as it is the most explored approach. Papers like Such et al[4] explore if the popularity of gradient descent approaches is of good reason or if other approaches can perform similarly, and have found genetic algorithms to be competitive in many different types of problems. This prompted me to try comparing a gradient descent approach with NEAT, an evolution strategy that evolves networks using a genetic algorithm.

The purpose of this paper at its core is to compare the initial transfer and the training after transfer when using a gradient descent approach as opposed to a non-gradient descent approach. DQNs [14] calculate the gradient of Q-value function via backpropagation, while NEAT mirrors nature in evolving small, simple networks that become increasingly complex to accommodate sophisticated problems.

The methods I am using in this paper have both been thoroughly researched, as well as many other aspects of transfer learning, which I will go through in the next section.

3. Introduction:

Transfer learning is a field with the potential to make using artificial intelligence significantly more efficient. Two methods of conducting transfer that seem to be conducive towards transfer are NEAT, an evolutionary algorithm, and Deep Q Networks, a gradient descent approach. Research towards these approaches is beneficial, since most research around transfer is focused on supervised learning. In order to get a complete view of the performance of transfer, evaluating it in multiple ways is important.

Transfer learning is a technique in which an agent learns to complete a problem and stores information about the problem in order to apply it to a different, but typically related problem. An example of this with humans would be learning the classic piano in order to learn the jazz piano or organ. It can reduce the amount of training required by using knowledge from other models and transferring it to similar problems. Since most companies deal with very similar topics, but may not be able to reuse the data in conventional models. This topic is important and can be pivotal in making artificial intelligence available to those with less available data or resources. Medical companies that may require painful tests in order to get data would want to maximize efficiency with their data. Transfer learning offers the potential to increase efficiency, allowing similar data to be utilized in multiple models. A relatively unexplored aspect of transfer learning is which methods of training enable transfer learning on different types of problems.

One training method that has potential to transfer well would be evolutionary algorithms utilizing population based approaches. Because these methods create a population of diverse artificial neural networks (ANNs) as solutions to the problem, a multitude of successful ANNs can be tested against a new problem, or the most successful ANN can be used alone. Another potential method to test is Q-Learning. This model is promising because Q-Learning can provide incentive for innovation when used in combination with neural networks, and there are examples of using Q-Learning to accomplish transfer learning in multiple fields (1).

Most current transfer learning research uses supervised learning and Bayesian networks (2). This is because supervised learning is already the most researched machine learning field, and transfer learning is essentially an extension of the method behind it as both are using prior training data to improve future performance. Also, the applications for transfer learning are typically in financial fields, potentially because the data is difficult to label, and the exact structure and conclusions to be drawn from the data may change often [1].

Transfer learning can be evaluated multiple ways. One way is to train an agent on the initial problem, then immediately evaluate it on another more difficult problem without any further training. I will refer to this as *initial transfer*, as it reflects how well the model transfers knowledge without any further training on the new problem. Another way that transfer is evaluated is by looking at the training on the target task after being trained on the initial task, which I will refer to as *training after transfer*. This measures how fast the agent learns after transferring.

In order to interpret our results from the experiments, it is important to have an understanding of the field in general, and the context of transfer learning and both methods in it.

4. Related Work:

Recently, strides have been made to increase efficiency, effectiveness, and innovate for parallelism in the field of neuroevolution, which utilizes a population of evolving networks. This is in part due to a paper published in 2002 that revolutionized neuroevolution, when the method of NeuroEvolution of Augmenting Topologies (NEAT) was developed. *Evolving Networks through Augmenting Topologies* [3]shows that NEAT increases efficiency due to crossing over different topological structures, protecting innovation through speciation, and beginning evolution in a minimal structure, and adding complexity through mutations. NEAT is crucial to the development of neuroevolution, and has been cited over 3000 times according to Google Scholar.

Determining what success looks like is important when conducting research on pre-existing models. One way to measure the success of a method is to compare it to the bonafide methods primarily used in the field. Clune et al. in their paper *Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning(2017)*[4] did that, noticing the heavy focus in the RL community on gradient descent approaches, and venture to see whether evolutionary algorithms can match the performance of more popular gradient descent approaches. The results show that certain circumstances, such as problems with local maxima, lend themselves very well to neuroevolution rather than gradient descent, although generally speaking population approaches are less computationally tractable. These results also imply that it may be better to conduct a dense search with neuroevolution in a neighborhood than follow the gradient. Another metric to measure success on a model is video games. Many older Atari games have distinct problems that researchers in reinforcement learning want to explore, such as SeaQuest being an optimization problem in which a policy that is too greedy will lead to a failed game. Lehman et al. in *A Neuroevolution Approach to General Atari Game Playing(2014)*[5] compare the performance of four different NE algorithms; conventional NE, CMA-ES, NEAT, and Hyper-NEAT; using three different state representations; object representation, raw-pixel

representation, and noise state representation. Noise state representation NEAT performed the best across the games under the first two state representations, which implies that adding additional complexity to solutions may actually hinder performance in certain situations. However, the geometric properties of Hyper-NEAT's connectivity allowed it to be the only approach capable of solving the noise state representation, which was the hardest as it is a single substrate of randomly seeded activations. What is even more interesting is that Hyper-NEAT performed better on the more difficult version of the problem, noise state, than the raw-pixel representation of Hyper-NEAT, which indicated Hyper-NEAT may have been overconstrained to the other two state representations. Also, the neuroevolution algorithms outperformed other common deep learning algorithms like standard State–action–reward–state–action (SARSA) [13] by quickly being able to find solutions to problems that trapped other algorithms in local maxima.

No matter what method is being used, it is important to determine if transfer will be applicable to real-world situations. In addition to financial and medical fields that were mentioned earlier, another practical example of transfer learning is the recognition of handwriting. Transfer learning is useful in the field because the amount of writing provided may not be enough to conclusively decipher the text. Baldominos et al. in *Hybridizing Evolutionary Computation and Deep Neural Networks: An Approach to Handwriting Recognition Using Committees and Transfer Learning* [6] evolved a population of CNNs to identify numbers from a handwriting database. The top 20 networks are chosen from the final neuroevolution population, then each network is trained more on the problem. From then on, the decisions are made on a "vote" by the council of the 20 chosen networks for what decision to make. The method did extremely well, outperforming nearly all of the previous work, like SARSA models on the chosen database, and even networks trained for letters outperformed some hand-designed tests trained for digits. Transfer learning has recently had research making especially big strides in efficiency, as that is a primary focus of transfer. Parlsotto et al. in *Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning* [7] introduce a technique to solve tasks and then transfer the knowledge with imitation learning

by training “expert” DQN networks for a specific task unique to each network. Once each expert network is trained, the Actor-Mimic Network (AMN) is trained from the expert networks. The AMN mimics the actions that the guiding networks would have produced. This maximizes efficiency by using networks that have already been made to guide the AMN.

Transfer learning can capitalize on neuro evolutionary methods, as the networks can create many networks using a variety of approaches, naturally leading to a large knowledge base by exploring many different solutions, which can be exploited on a range of tasks. Doncieux in *Transfer Learning for Direct Policy Search: A Reward Shaping Approach* [8] proves the effectiveness by exploiting a knowledge base gathered by using neuroevolution on a simpler task. Analysis of the solution on the more difficult task showed that the knowledge base captured important features from the simpler task. Transfer learning also has different ways of being carried out. *Transfer of Driving Behaviors Across Different Racing Games* [9] explores the most effective and efficient ways, between simply copying a network’s knowledge, copying the learning mechanism used to evaluate, or a combination of the two. The paper found that the latter performed best due to being able to adapt most flexibly.

Little work has been reported on the cases for which transfer learning is not effective. *Visual Analogies Between Atari Games for Studying Transfer Learning in RL* [10] does that, exploring the relationships of transfer between different atari games, trying to find the pros and cons of transfer. One reason transfer may perform poorly in practice is that two logically similar tasks may need to be approached very differently by a model.

Another paper exploring transfer learning investigates a novel way to transfer. *Visual Transfer between Atari Games using Competitive Reinforcement Learning* [11]takes advantage of similarities with the images in video games to transfer knowledge, resulting in improvements in both efficiency and performance. However, a difficulty in emphasizing efficiency is that diversity can be lost in the process. This paper notices that a problem with creating solutions that transfer well is that it is difficult to bias networks toward modularity without restricting the search, or preventing an optimal solution. The paper’s

approach is to begin with a bias towards modular solutions that is phased out in later generations. The results are that applying this technique to Hyper-NEAT achieves a 91% success rate on a problem that the standard Hyper-NEAT only achieves a 12% success rate.

A problem with one network learning multiple tasks is that catastrophic forgetting may arise. This is when the ANN loses the ability to perform a previously learned task upon learning a new one. Clune et al. wrote *Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills* [12]trying to address the problem of catastrophic forgetting by creating a more systematic way to create modular neural networks. This would help solve catastrophic forgetting, as learning could be turned on or off based on whether the model needs to learn the task at hand. The approach used is to assign costs to connections in the network, to encourage modularity. The results show that the modular networks learn faster, retain more, and perform better than standard networks.

There are multiple definitions of modularity including the networks themselves or specific nodes of the network. Neural networks are generally favored when solutions require a multi-faceted approach, and this paper investigates for what type of tasks are modular networks favorable. The results indicate that problems emphasizing accuracy rather than speed, such as medical information and flight ordering, are more conducive to modular neural networks, and modular problems are more easily solved with modular neural networks due to the ability to adapt.

In the next section, I will discuss the ways I designed my experiments, and the thought process that went into preparing the environment and comparing the performance of two fundamentally different methods.

5. Methods:

The first thing I needed to do was be able to ensure I could actually perform experiments without burdening myself with enormous training times, so I had to do preprocessing of the environment. I also had to make sure transfer would actually be meaningful by making a different enough environment from Car Racing V0, so I added junctions, varying track width, and obstacles to the course. Also, since the models I am using are so different, it is important to ensure that the comparison will be fair to both methods.

5.1 Preprocessing

Before any agents performed either version of car racing, I wanted to reduce the size of the input images. This is because there are many aspects of the game that would simply take up more space. That could limit experimentation due to making simulations extremely cumbersome in terms of memory and computational power. The first thing I did was convert the image into grayscale, which reduced the computational load without affecting performance of agents, as the track is dark, and the grass is light. Also, the shading was completely removed, so the track was all one shade, and the grass was all one shade, rather than the multiple colors of grass. The last thing I did was something inspired from David Silver's work on Atari Games in "Human-Level Control through Deep Reinforcement", where he stacked the consecutive frames to reduce the number of frames needed to train. I did something similar, stacking 4 frames at a time.

5.2 Meaningfully Different Task

In order to evaluate transfer learning on NEAT and DQN, there needs to be a problem similar enough to provide opportunity for a knowledge base, but different enough that the agents aren't simply repeating the task that it was trained on. The car racing environment on OpenAI Gym is easily editable, so adjustments were made to the formulation of tracks in order to make the problem more difficult for the agents to solve. The changes I made to the original Car Racing V0 (V0) were adding junctions, which is simply a split in the track, varied track width, and obstacles.

5.2.1 Junctions:

Junctions were added to the tracks in order to force the agent to make choices. I originally thought that this would be the most impactful change to the track in terms of increasing the difficulty for agents to complete, especially for a DQN. This is because before, the track was a straightforward path and the agent did not need to account for decision making, only figuring out the path of the track through increasing rewards. The idea of the changes would be to make it significantly more difficult for an agent to realize that it is not on an optimal path, especially for the DQN, since a common downside of gradient descent approaches is being caught in local maxima. However, the results showed junctions were not particularly more challenging than the next two challenges, perhaps due to the relative poor overall performance of the agents.

This image shows the modified version of the track with junctions added.

5.2.2 Varying Track Width:

Changing the track width throughout the course from the original width in v0 effectively forces the agent to be more precise. The wide track in v0 is relatively forgiving for “reckless” driving as long as you remain on the blacktop - changing the track width will likely result in lower scores on earlier generations and episodes because it is far more likely that the agent will spin out each run.

This image shows a version of the track that has been modified to feature varying track width.

5.2.3 Obstacles:

Adding obstacles to the course in v0 accomplishes a similar goal to varying the track width. However, it is significantly easier for the agent to recover when going off track briefly than hitting an obstacle. The agent can even effectively ignore the varied track width on occasions like above, where cutting a corner will not likely force the agent to spin out. However, with the obstacles, there is much less room for error. Hitting an obstacle results in the car coming to a stop, and the agent likely needing to back up, effectively ending the run. This is a way of enforcing the goal of the above in a different way.

This image shows a version of the track that has added obstacles.

5.3 Justifying comparison

The nature of the methods I am using may lead to questions regarding the feasibility of comparing them. This is a valid concern, as one might point out that NEAT will of course perform better, as its population based approach will more likely produce a better network, if for no other reason than it has more networks attempting. In order to create a more complete picture surrounding the comparison, I have taken an approach that incorporates multiple facets of the approaches.

The goal of transfer learning is not simply to attain the maximum possible score on the new task. Other goals like training time saved and how much you learn from the original task must also be considered. To attain a realistic view of transfer, we cannot have unlimited training, as transfer learning would be used in situations where training data may be sparse. I have performed three experiments to attempt to get a broad comparison of the two approaches, as well as a deeper look into experiment 1.

Experiment 1's purpose is to get a baseline of a comparison, with no attempt at making a fair comparison, even though that likely means NEAT will look much better. In experiment 2, the goal is to observe the training after transfer, with the models only trained to 350 on V0. In experiments 3, the goal is to observe the initial transfer as well as observe the effect of the challenges added. Again in experiment 3, the models are trained to at most 350 on Car Racing V0. The reason the models are trained to at most 350 on V0 is because the DQN's score only ever reaches about 400 on V0, so in order to make a more fair comparison, I have cut off the learning at 350 to ensure they both have the same knowledge base of the initial problem.

For NEAT, the population size was 100, with 100 generations being performed. There is no fitness threshold for experiment 1, but for experiments 2 and 3, the threshold is set to 200 and 350. The DQN was a fully connected network with 5 output nodes, for each action.

6. Results:

6.1 Experiment 1:

Complete training on CarRacingV0 using both approaches. Using those pre-trained networks as a starting point, continue training on CarRacingV1 and find the maximum score for each approach. The goal of the experiment is to make a straightforward comparison between the two with no limitations on performance to make it fair, because NEAT is expected to perform better due to running many more simulations.

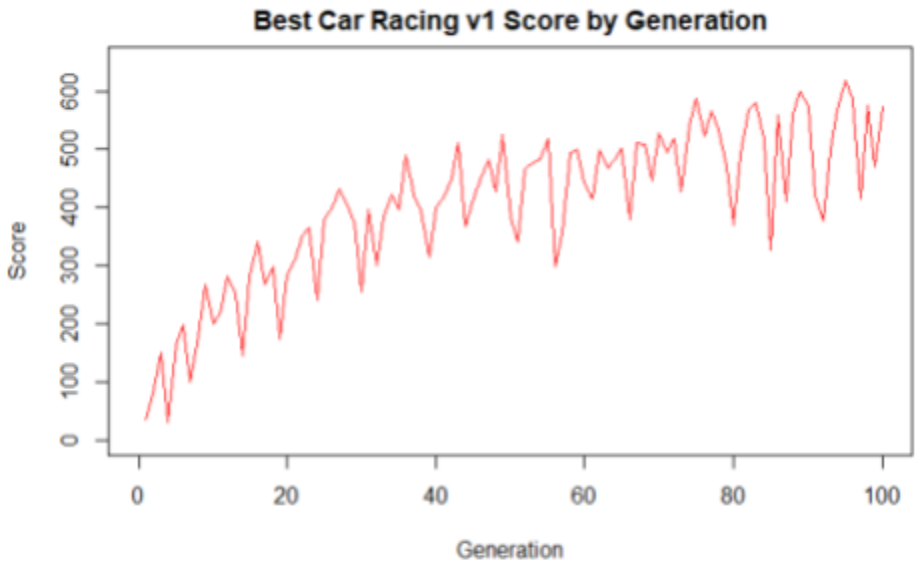

Figure 1: NEAT's performance after 100 generations of learning onto Car Racing V1 after learning to a score of 350 on Car Racing V0.

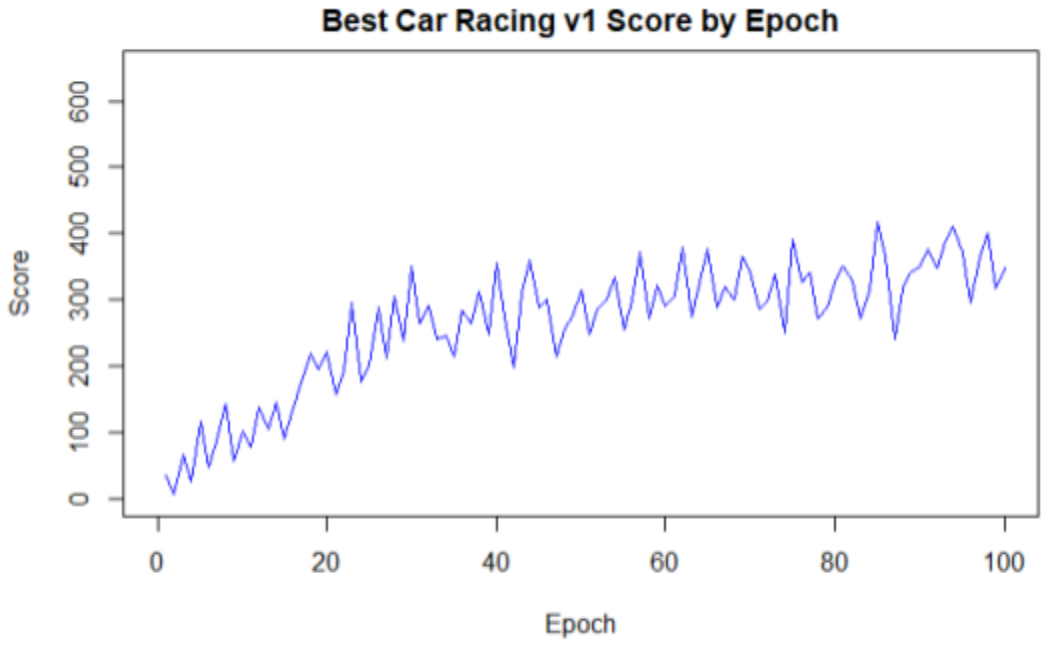

Figure 2: DQN's performance after the same process from Figure 1.

The figures above illustrate that NEAT, unsurprisingly as NEAT performs many more simulations due to the population approach, performs significantly better after an equal number of generations to epochs.

6.2 Experiment 2:

Experiment 2 served the purpose of observing the speed of training after transfer. These figures will compare how much training is needed to achieve a score of 350 on V1 after training to a certain score on V0, with the maximum being 350 on V0.

Method	Max Score on V0	Generation/Epoch	Simulations ran
NEAT	None	38	3800
NEAT	200	27	2700
NEAT	350	21	2100
DQN	None	45	450
DQN	200	34	340
DQN	350	30	300

Figure 3: The table shows the score trained to on V0, the generation or epoch needed to get to a score of 350 on V1, as well as how many total simulations the agent would have ran to get to that point.

This table highlights the differences between the methods, showing how different the number of simulations per generation or epoch is, and why an effort is needed to ensure the comparison is fair.

6.3 Experiment 3:

The goal of experiment 3 was to compare the *initial transfer*. The reason it was split up into two parts, a and b, was to compare initial transfer in two ways. Experiment 3a compares the *initial transfer* of each

method on V1 after having trained to a maximum of 350 on V0. In other words, experiment 3a trains an agent on V0, and evaluates it directly on V1. For experiment 3b, the track was divided into components that I added in the beginning, which was the junctions, varied track width, and obstacles. Each agent trained on V0 would do the tracks separately, to see if a particular method was better or worse at handling the different type of challenge, or if a particular challenge added more difficulty than the others.

6.3.1 Experiment 3a:

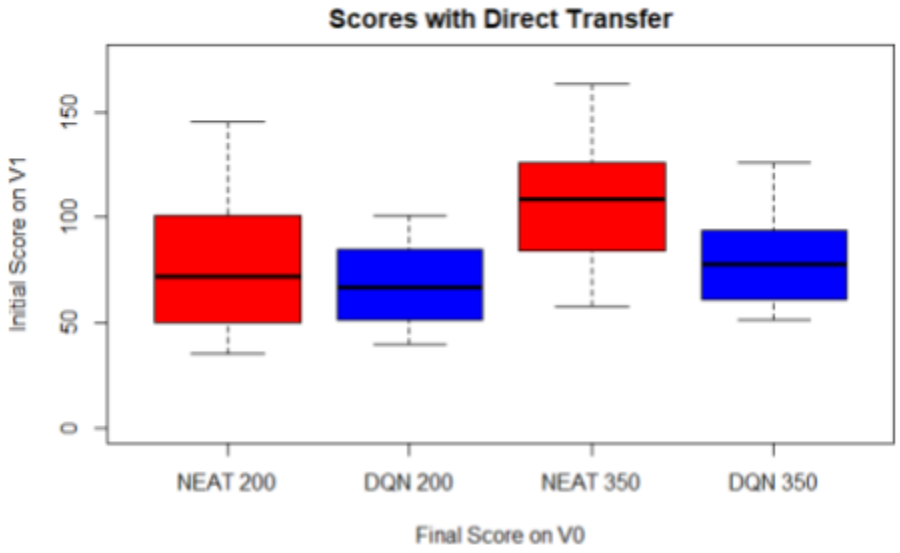

Figure 4: The boxplot above shows the range of scores with direct evaluation on V1 after the agents were trained to a score of 200 on V0, as well as agents trained to a score of 350.

In Figure 4, the X-axis shows which method is used and the score the agent trained to on V0. The Y-axis shows the score of the evaluation. T-tests were conducted to see if there is evidence to conclude there is a substantial difference in transfer between NEAT and DQN. DQN vs. NEAT (200) reports a p-value of .083, and DQN vs. NEAT (350) reports a p-value of .070. Both are very close to achieving statistical significance.

6.3.2 Experiment 3b:

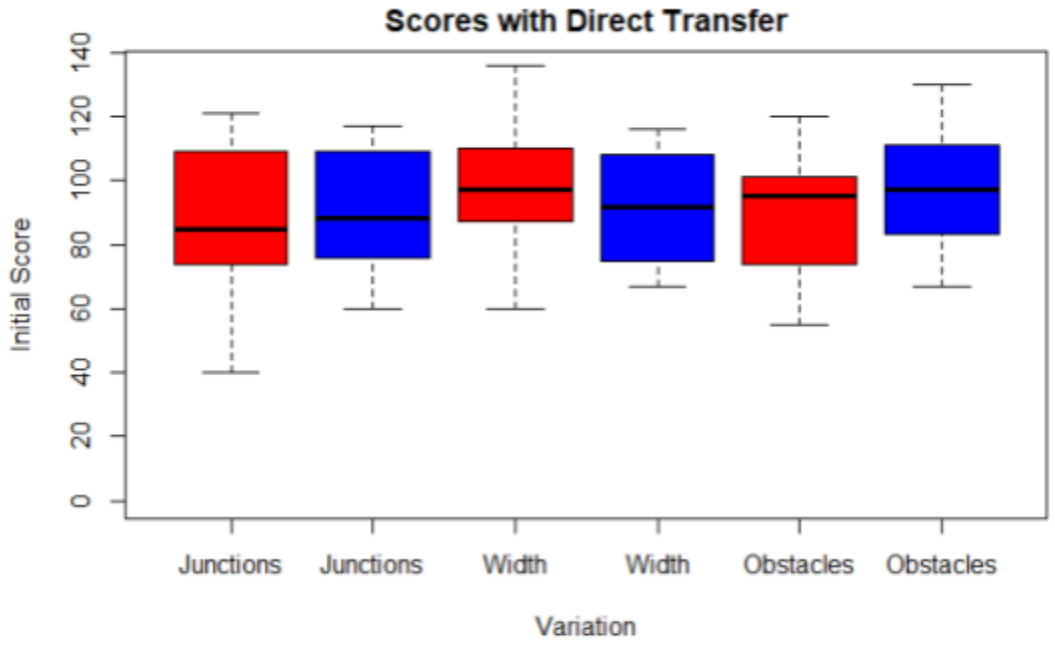

Figure 5: The boxplot above shows the range of scores with direct evaluation on tracks with only one of the challenges. NEAT results are colored in red, DQN results in blue.

T-tests were also conducted with this experiment to see if there is evidence to conclude there is a substantial difference in transfer between NEAT and DQN on the different challenges. DQN vs. NEAT (Junctions) reports a p-value of .56, DQN vs. NEAT (Varied Track Width) reports a p-value of .41, and

DQN vs. NEAT (Obstacles) reports a p-value of .53. None are close to statistical significance, which can show that none of the challenges are significantly more difficult than the rest.

6.4 Further Analysis on Experiment 1:

An interesting topic is not just how well the agents transfer, but how quickly the agents learn while transferring. A look into the first stages of transfer may give insight to this, as an agent that learns significant amounts in the beginning stages can be very useful for efficiency.

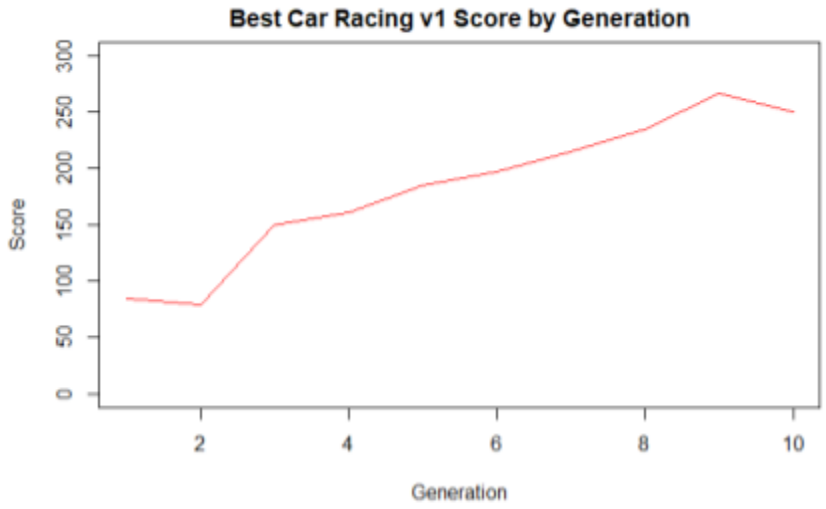

Figure 6: The plot shows the average score across the population of the training after 10 generations of NEAT. The highest average score in the first 10 generations is 261.

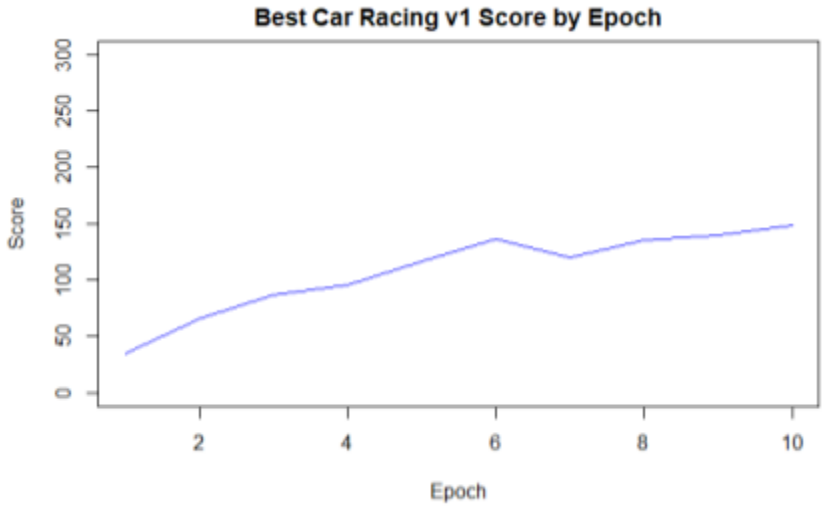

Figure 7: This plot shows the average of 10 trials of the DQN after 10 epochs of training. As shown in the plot, DQN accomplishes significantly less learning in the first 10 epochs than NEAT does in the first 10 generations.

This section's purpose was to only report the results and framework of my experiments. In the next section, I will interpret the results and the outcome of the experiments.

7. Discussion:

The results of my thesis are not strong enough overall to make a conclusion about which method is definitively better at transfer on my specific problem. In experiment 3a, the data I gathered showed a strong trend, despite being statistically insignificant. I believe if the sample size of experiments was larger, the p-value to attain statistical significance would be achieved due to the strong trend that was shown despite the small sample size. Also, with experiment 3b where the difficulty of the challenges were compared (junctions, varied track width, and obstacles), the lack of any significant trends can be

taken as suggesting a conclusion about the lack of differences. Although I thought that the junctions would be significantly harder than the other two challenges, perhaps the reason it wasn't is that the agents are not high performing enough to make decisions around the track. Like I mentioned earlier, the score in order to consider the track 'solved' according to OpenAI gym was a score of 900, so the junctions likely would have posed a higher difficulty for those agents, which already possessed a high aptitude for maneuvering the track, where my agents had not perfected the mechanics.

The results from experiment 2, which compared the speed of training after transfer, show that if high efficiency is necessary, and it is not feasible to run many simulations at once, DQNs would be desirable. NEAT requires many simulations to be run so that the population can be trained. Although DQNs don't reach the same score in as many epochs, each epoch takes significantly fewer simulations to actually complete. DQN would be an excellent choice for smaller scale work and those without access to high performance computers.

Experiment 3a, which compared the initial transfer onto V1, gives different insights. NEAT scores much better with direct transfer than the DQN. Observing this, as well as using the analysis from the first generations of transfer from experiment 1, the overall picture becomes more clear. NEAT performs very well with direct transfer and small amounts of training. This shows NEAT might be more useful for situations where it is not possible or easy to perform transfer onto the new task, or there is little available time or data for training. NEAT will make use of just a few generations and learn a lot initially, as well as performing well with no training on the target task. However, combining this knowledge with what we know from experiment 2, the more training that can be done on the target task, the more feasible using DQNs becomes. More research would be needed to determine if and when a switch from NEAT to using a DQN would become more efficient for the performance gained.

A significant amount of the work done in this paper was attempting to ensure fair comparisons between the two methods. Since a population based approach can be difficult to compare to gradient descent based methods simply because of the number of simulations and networks being trained, it is not

a straightforward task to compare these. This paper tackles that problem and hopefully inspires those who continue research in this area to not be discouraged by the comparison of two fundamentally different strategies.

The work that I did was complete, but there are many branches that could be followed up on in future research which I will discuss in the next section.

8. Future Work:

This paper opens up several avenues for future work. In my opinion, the most interesting possibility for further research would be trying this comparison on different types of problems. Historically, DQNs seem to not perform well on Car Racing V0. This could be because the size of the network needed is very large. A problem with a smaller state space and a discrete action space may be easier due to it being easier to process in a smaller network. The results may be more competitive, and fewer restraints needed to be added to make the comparison fair, since the potential for learning would ideally be closer.

Another topic I see as worthwhile of more research would be focusing more on the population aspect of NEAT in regards to Car Racing V1. Specifically, one could take the top 10% of the population and try running them each individually on the tracks that only had one challenge. That way, we could see the potential advantages of the population effect on learning, as it could be investigated if the agents explored the junctions differently than the varied track width or obstacles. In the interest of keeping the comparison fair, many of the aspects of NEAT that make it an interesting method were not investigated, such as speciation, but further research could provide additional insights into its potential.

9. Conclusion:

This paper explores how well two fundamentally different methods of learning compare when used for transfer learning, one population based approach in NEAT, and one gradient descent based approach in DQN. The transfer was done using Car Racing V0 from OpenAI Gym as a benchmark for starting transfer and modified versions of Car Racing V0 for transfer testing. The transfer was evaluated in terms of overall performance without any restrictions after training on V0, initial transfer after training on V0, and training after the initial transfer after training on V0. When performing initial transfer and with little training on the target task, trends show NEAT performs better although without statistical significance. When the training on the target task is more substantial, the results are less clear, as NEAT performs well but the tradeoff for the efficiency of DQN needs additional analysis. These results show that transfer learning can be viable on multiple methods, and different situations may call for different approaches. One of the significant contributions of this work is showing that comparisons between population based approaches and gradient descent are feasible and provides one such approach. Opportunities for future work include comparing a population based method to a gradient descent method on different problems, as well as further exploration of using the population of NEAT to see if different parts of the population solve different parts of the problem. Transfer learning can be viable with approaches other than supervised learning, and further research into innovating transfer can make it easier to use for those without the resources for large datasets or expensive computational setups.

10. Acknowledgements:

First and foremost thanks to Jason Yoder, my thesis advisor. Always ready with productive feedback, ideas on how to move forward and improve existing work, and a positive attitude when I was facing difficulty.

Also, I would like to thank my family, specifically my mother, Lucinda Dunaway. Frequent texts of “Hey sonny” and constant inquiries about the state of my thesis were not unappreciated, as well as the unconditional support.

Next, I would like to thank my girlfriend Maggie Mirabelli. Steady support and willingness to listen to some complex ideas with no background knowledge was extremely useful and always welcome.

Others I would like to thank are included but not limited to Gabe, Damon, Seth, Spencer, Ishan, Momo, Jared, Shipley, Traven, Matt, Megan, Eli, and Bryce.

11. References:

[1] Gyeeun Jeonga, Ha Young Kima. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. Ajou University, 2019.

[2] Karl Weiss, Taghi M. Khoshgoftaar, DingDing Wang. A survey of transfer learning. Journal of Big Data, 2016.

[3] Kenneth O. Stanley, Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. University of Texas-Austin, 2002.

[4] Such et al. Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. Uber AI Labs, 2018.

[5] Matthew Hausknecht et al. A Neuroevolution Approach to General Atari Game Playing. IEEE, 2014.

[6] Alejandro Baldominos et al. Hybridizing Evolutionary Computation and Deep Neural Networks: An Approach to Handwriting Recognition Using Committees and Transfer Learning. Complexity, 2019.

[7] Emilio Parisotto et al. Actor-mimic: Deep multitask and transfer reinforcement learning. University of Toronto, 2016.

[8] Stéphane Doncieux. Transfer learning for direct policy search: a reward shaping approach. Proceedings of ICDL-EpiRob conference, 2013, Osaka, Japan.

[9] Luigi Cardamone et al. Transfer of driving behaviors across different racing games. University of Illinois at Urbana-Champaign Genetic Algorithm Laboratory, 2011.

[10] Doron Sobol et al. Visual Analogies Between Atari Games for Studying Transfer Learning in RL. University of Tel-Aviv, Facebook AI Research, 2018.

[11] Akshita Mittel et al. Visual Transfer between Atari Games using Competitive Reinforcement Learning. Robotics Institute, Carnegie Mellon University, 2019.

[12] Kai Olav Ellefsen et al. Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills. PLOS Computational Biology, 2015.

[13] Harm van Seijen et al. A Theoretical and Empirical Analysis of Expected Sarsa. IEEE, 2009.

[14] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. Springer Science, 2015.

[15] Greg Brockman et al. OpenAI Gym. arXiv, 2016.