Impact of Changing Transition Function in Deep RL

Nicholas Lopez

May 7, 2024

Abstract

In this project, we explore the Out-of-Distribution generalization problem applied to Deep Q-Learning agents. In this problem, agents are exposed to testing environments or data different from the ones in which they are trained. This problem thus has countless real-world applications, from self-driving vehicles to automated factory machinery. Through the experiments, we see the importance of exposing agents to various training conditions and evaluate their performance when exposed to each condition when testing. To examine this, we train agents using Deep Q-Learning to play Pac-man in environments with varying ghost behavior and independently test them under each of these different conditions. The hypothesis that more hostile training results would always lead to better performance overall was incorrect. In particular, we see that an agent trained among the most complete balance of aggressive and randomly-moving ghost behavior performs the best, as it is likely the most representative environment for the overall set of test conditions. This agent had an aggregated win rate of nearly 70%. Meanwhile, the agent trained among ghosts with the most "monotonic" behavior perform worst overall when exposed to the various environments, with an aggregated win rate of less than 52%. Through these experiments, we see that exposing a Deep Q-Learning agent to the most hostile conditions dues not always best prepare an agent for overall performance across a variety of other conditions, even less hostile ones. This is because agents may focus on less important, more environment-specific details rather than the general interactions of the environment. Similarly, we see that an increase in the layer count of a DQN prepares an agent for more complex environments, while lower layer counts more suitably fit agents in environments where it is preferable to avoid overfitting or focusing on details that are irrelevant in other, yet still important environments.

1 Introduction

1.1 Motivation

This paper explores the Out-of-Distribution (OOD) generalization problem as it pertains to Deep Q-Learning agents. In the OOD generalization problem, an agent is tested using data or an environment that is different from that upon which it was trained. Various effects, such as the quantity of training time/data and exact environment/data upon which the agent was trained, determine how well the agent performs in the unique tests.

In Deep Q-Learning, the agent's states are given as input to a Deep Q-Network (DQN), which returns Q-values for each actions, representing approximate benefit of selecting that action. Through training, the DQN can approximate the Q-values with increased accuracy. Thus, during testing and when implemented, the agent selects the action with the highest Q-value.

Thus, the OOD generalization problem, especially as it pertains to Deep Q-learning, is increasingly applicable. For example, imagine a self-driving vehicle trained for driving within a particular city. If, during testing, the vehicle is placed in a different city, it is important the agent can perform effectively in the new environment, quickly adapting to any changes in the environment, ranging from speed limit changes to more nuanced differences such as driving norms in that particular region. Similarly, imagine an unmanned, exploratory or recovery-focused undersea drone trained in one particular region of the ocean. If the agent is needed in another region, perhaps with different undersea currents or wildlife behavior, it is important the agent is able to quickly and effectively adapt to this new environment. There are countless applications for Deep Q-Learning, and bolstering the efficiency of such agents within the OOD generalization problem will be increasingly pertinent in the world of machine learning and, eventually, daily life.

1.2 Question & Hypothesis

To examine this problem, we trained a Deep Q-Learning agent to play the game Pac-Man, modified to be played on a slightly smaller layout. To create differences in the training and testing environments, we altered the ghost transition functions. In particular, four classes of ghost agents were created, each of varying levels of aggression and awareness. Furthermore, various layer counts were used for the DQN, as we aimed to examine potential differences in performance across this axis. Thus, while we focus primarily on the former, our research question is two fold. We examine how changing ghost movement probabilities to be different in testing than it was in training affects the performance of the reinforcement learning agent. We further examine how the number of layers used affects agent performance within across the changes in transition function.

We expect the agent's performance to decrease as the probabilities deviate further from those used in training. In particular, we expect the agents trained in the more hostile environments to perform better than those trained in more accommodating environments when exposed to conditions other than those experienced during training. An agent trained in a more hostile environment will be better prepared to evade and attack ghosts during the proper times and will also be better equipped to traverse the provided landscape. Meanwhile, an agent trained in a more friendly environment will be less aware of the need to evade ghosts when vulnerable, seeing them less as threats due to their previously less aggressive behavior. In general, we expect performance to increase as the number of layers in the DQN increases, as the representational and computational abilities are increased to more accurately determine the Q-value of each action. Furthermore, we expect performance increases to be more significant with changes in DQN layer count for agents trained in more hostile environments, where these abilities are increasingly more important, as the agent is less likely to find success without being aware of the intricacies of the game.

2 Literature Review

Because of the significance and applicability of the topic, there is plentiful literature discussing the problem and potential solutions. The following literature was particularly helpful in motivating the specific methodology and procedures used, while also providing insight into the current state of applicable research.

- *Reinforcement Learning in Non-Stationary Environments* by Padakandla et al. (2019) discusses the need for Reinforcement Learning models that can adapt to constantly changing environments, from traffic control to robotics. To optimize agent performance, the paper proposes a new method called Context Q-Learning that continually learns, optimized for dynamically changing environments. This new approach uses a specific algorithm to detect changes in an environment and can thus cause the agent to improve its actions more quickly and adapt to the new environment more quickly.

- *A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments* by Padakandla (2020) discusses both the need and current benefits and drawbacks of using Reinforcement Learning algorithms tailored to dynamically varying environments. In particular, it notes that the exploration of this specific problem is relatively young, and there remains significant progress to be made.

- *Goal Misgeneralization in Deep Reinforcement Learning* by Koch et al. (2021) discusses one type of out-of-distribution robustness failure. It discusses the causes of such failures and the difference between such failures and capability generalization failures.

- *Contextual Q-Learning* by Pinto and Vale (2020) discusses one such approach to the problem at hand. In particular, the model learns to estimate the true outcomes and rewards given the current information. Depending on the following outcome, the model can adaptively change its learning rate under the assumption that its current environment may have changed.

- *The Role of Pretrained Representations for the OOD Generalization of RL Agents* by Dittadi et al. (2021) discusses ways in which Reinforcement Learning agents can be trained to perform effectively in altered environments. Specifically, simplified representations of an environment may enable agents to perform more optimally in larger, more complex versions of the environment.

These papers show the importance and prevailing need for further research in the Out-of-Distribution generalization problem.

3 Methodology

We conduct the experiments in three distinct phases to most effectively explore each problem. The initial codebase is from van der Ouderaa (2016), which contained the environment and gameplay code for Pac-Man, as well as the initial framework for a 4-Layer DQN. In subsection 3.2, we detail the modifications made to the codebase.

3.1 Key Papers

Discussed previously, *Goal Misgeneralization in Deep Reinforcement Learning* by Koch et al. (2021) and *The Role of Pretrained Representations for the OOD Generalization of RL Agents* by Dittadi et al. (2021) will be relevant to the paper. The first paper discusses ways in which Reinforcement Learning models behave without maximizing the reward function, which is definitely a potential result in the experiments, especially with the agent being tested in an altered environment. The second paper implies a key way in which agent performance may be maximized. Perhaps the environment that represents the most generalizable version of the Pac-Man game will be most optimal for training agents.

3.2 Code & Experiment Preparation

The code used for the project is provided in the GitHub repositoty available here (`https://github.com/nrlopez03/neuro240-final`). Three key extensions were implemented for the experiments.

3.2.1 Ghost Transition Functions

The first change implemented was to create two new classes of ghost agents. Thus, the experiments were run using the four following classes:

- Right: This ghost agent will always move right when possible and move in a uniformly random direction when not possible.

- Random: This ghost agent will always move in a uniformly random direction from those possible.

- Directional: When Pac-Man is vulnerable, this ghost agent will move in a direction that minimizes distance between itself and Pac-man with probability 0.8 and move in a uniformly random direction from the remaining available directions with probability 0.2. When the ghost is vulnerable, this ghost agent will move in a direction that maximizes distance between itself and Pac-man with probability 0.8 and move in a uniformly random direction from the remaining available directions with probability 0.2. Thus, apart from edge cases, such as when the ghost or Pac-Man's vulnerable is about to expire, this class of ghost can be thought of as one that makes the optimal move with probability 0.8.

- MaxDirectional: When Pac-Man is vulnerable, this ghost agent will always move in a direction that minimizes distance between itself and Pac-ma. When the ghost is vulnerable, this ghost agent will always move in a direction that maximizes distance between itself and Pac-man. Thus, apart from edge cases, such as when the ghost or Pac-Man's vulnerable is about to expire, this class of ghost can be thought of as one that always makes the optimal move.

While the Random and Directional ghosts were provided in the codebase, the Right and MaxDirectional ghosts were created to expand the range of aggression across the environments. In the Right ghost class, a less hostile, more exploration-friendly environment is created that allows the Pac-Man agent to navigate the landscape more easily. However, as hypothesized, it may also prevent the agent from properly learning the nuance of the game and ways to avoid ghosts, a potential roadblock in later experimentation with more hostile ghosts. In contrast, the MaxDirectional ghost class was created to be as hostile and aggressive as possible. When trained in this environment, is is hypothesized that Pac-Man will more effectively learn to evade ghosts, yielding higher performance during testing with easier, less aggressive ghosts.

3.2.2 Creation of New DQNs

While a 4-Layer DQN was provided in the codebase, ones of 3 and 5 layers were also created for experiment purposes. While the 4-Layer DQN contains two convolutional layers and two fully connected layers, the 3-Layer DQN contains two convolutional layers and one fully connected layer, and the 5-Layer DQN contains three convolutional layers and two fully connected layers.

3.2.3 Efficiency Enhancements

Finally, enhancements were made to make experiments more operationally efficient. Printouts were reduced when needed, and the code was changed to write the results to a CSV file after experimentation. Parmaters were added to track the number of layers, training agents, testing agents, and number of training games used for the given experimentation. Furthermore, the command line argument structure was altered to allow for the ghost classes to switch between training and testing for a given agent.

3.3 Initial Tests with 4-Layer DQN

Following the completion of the necessary preparation, experiments were made to determine initial performance to create baseline metrics. Each agent was trained in a different environment, each containing ghosts of different classes, for 5,000 training games using the 4-Layer DQN. Then, the agents were tested independently for 1,000 games in each of the four environments.

3.4 Testing with DQNs of Various Layer Counts

Building upon the previous results, we expand the experiments to test the number of layers in the DQN as a variable. Thus, we create agents trained with DQNs of 3, 4, and 5 layers. Each agent is trained in one of four environments before then being independently tested in each of the four environments for 1,000 games each. This allows us to examine how changing the number of layers in the DQN affects an agent's ability to adapt and respond to an altered environment.

3.5 Testing for 50,000 Training Games

Finally, we tested a set of agents for 50,000 training games. For each training environment, we picked the DQN layer count that yielded the best overall performance. Using this DQN, we trained an agent in each of the given environments for 50,000 games each. We then independently tested each agent for 1,000 games in each of the four environments.

4 Results

We evaluate our results in the order they were obtained, analyzing each step.

To gauge agent performance, we rely upon two key evaluation metrics, game score and win rate. The primary evaluation metric (and, thus, the focus of the DQN) is the score of each agent while playing the game. As it is naturally part of the provided Pac-Man environment, it will be standardized throughout the experiment. Score naturally decreases at every time step, but increases when Pac-Man eats dots located around the map or consumes a ghost when it is possible to do so. Thus, the agent is incentivized to both navigate the map safely, consuming dots and ghosts in the process, but also be mindful of the time required to do so. The second metric is win rate, directly correlated with score and often a useful proxy given its normalization. Pac-Man loses the game when attacked by a ghost while vulnerable but wins by consumin all dots before being attacked.

4.1 Performance for 4-Layer DQN

We first examine the performance of the agent with an underlying 4-Layer DQN, exposed to different conditions when training. Figures 1 and 2 show the average scores and win rates of the agent when exposed to varying environments for 5,000 training games and then tested for 1,000 games in each environment.

We see that the results contradict the hypothesis. Under these conditions, the agent trained in the environment with Right ghosts perform best, followed by the Random, MaxDirectional, and Directional ghosts. While these results are surprising at first, the explanation is quite simple: with the given amount of training games, agents trained under easier conditions survive longer. Thus, they are able to learn to navigate the environment and learn the game more effectively. Meanwhile, agents trained in the more hostile environment die quickly, before they are able to learn the intricacies of the game and to effectively navigate the environment.

We can also draw insights by examining across test type. In particular, as expected, the tests in the environment with the MaxDirectional class of ghosts yield the worst performance for each agent. Similarly, the Random and Right classes of ghosts yield the strongest performance results for each agent, with the Right class yielding the best performance for 3 out of 4 of the agents. These results make sense, given the hostility and aggression of each ghost class.

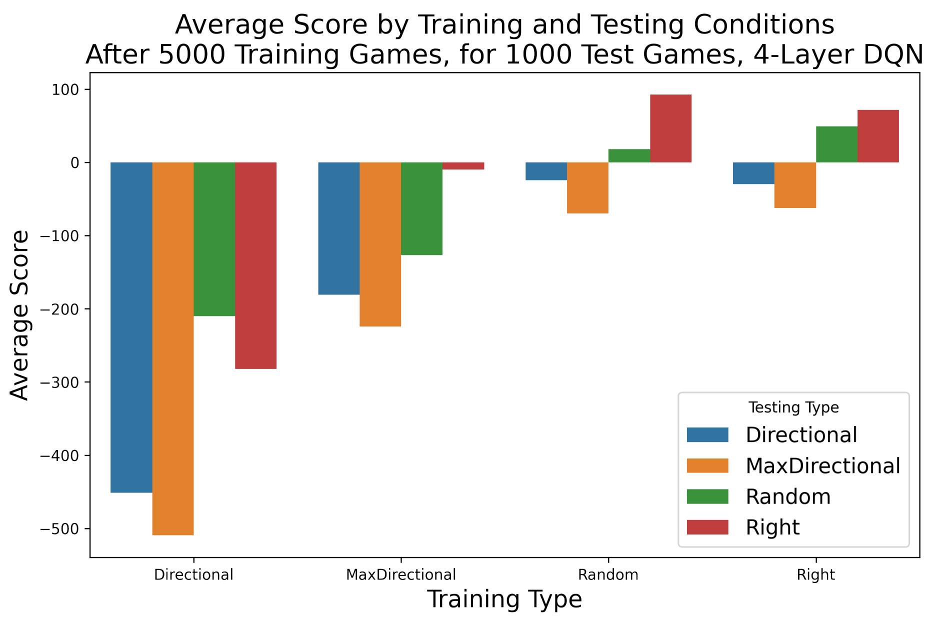

Figure 1: Average Score for 4-Layer DQN by Training and Testing Conditions

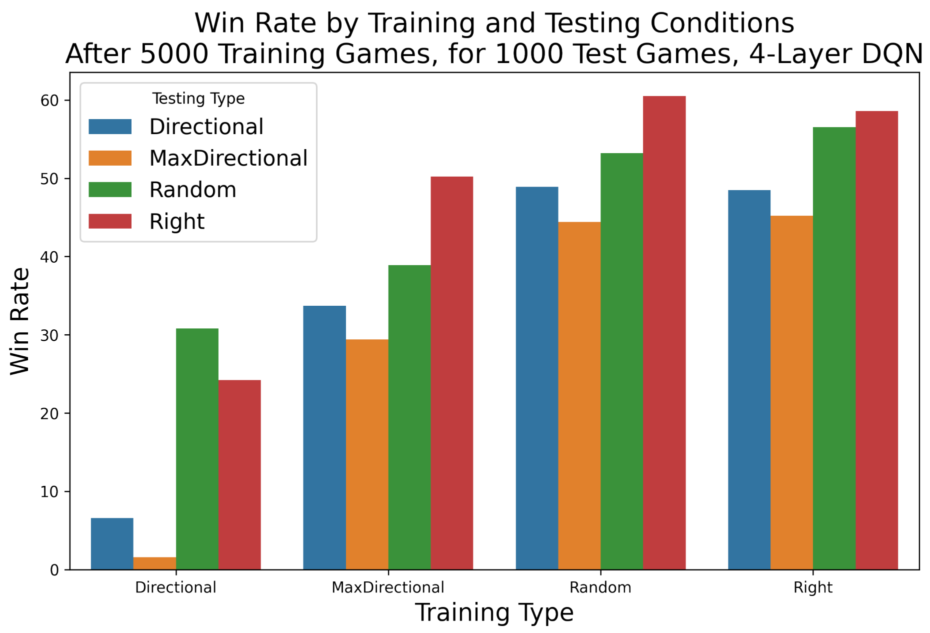

Figure 2: Win Rate for 4-Layer DQN by Training and Testing Conditions

4.2 Performance for DQNs of Various Layer Counts

We then test across various training environments and layer counts for the DQN. Figures 3 and 4 show the average scores and win rates of the agent when exposed to varying environments for 5,000 training games and then tested for 1,000 games in each environment, by both training environment and layer count in the DQN.

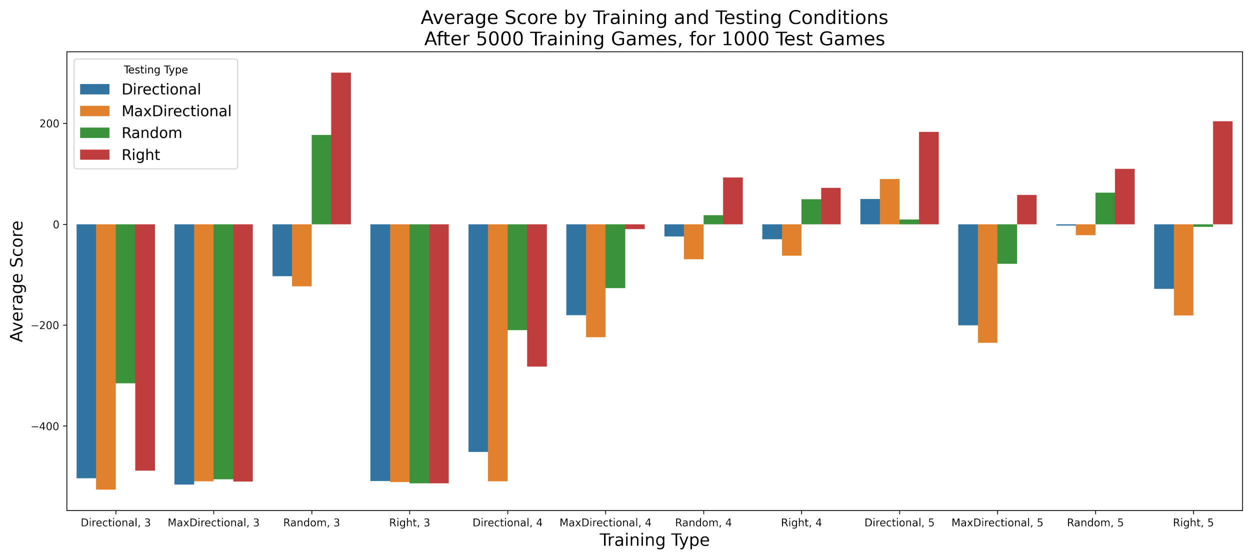

Figure 3: Average Score for DQN of various layer counts by Training and Testing Conditions

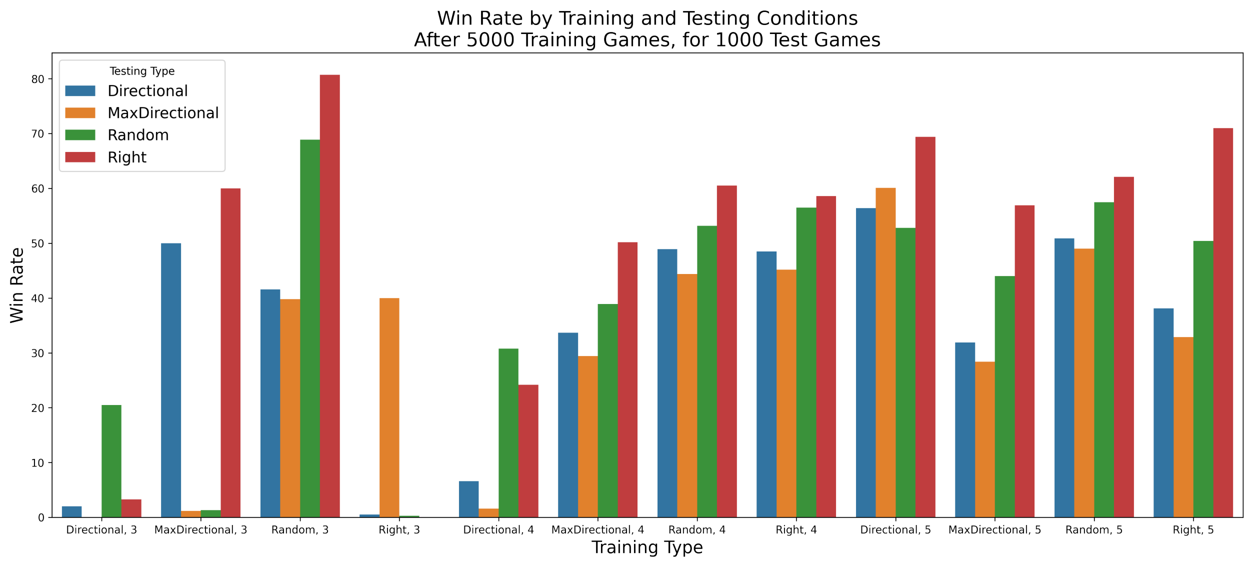

Figure 4: Win Rate for DQN of various layer counts by Training and Testing Conditions

As expected, for the agents trained in the environment with the Directional and MaxDirectional classes of ghosts, overall performance increases significantly as the layer count in the DQN increases. In particular, for the agent trained amongst the Directional ghosts, average win rate increases from 6% to 16% when increasing layer count from 3 to 4, and to a significant 60% when increasing layer count to 5. Similarly, for the agent trained amongst the MaxDirectional ghosts, average win rate increases from 28% to 30% when increasing layer count from 3 to 4, and to 40% when increasing layer count to 5.

These results make sense. More layers allow for more representational power, increasing the agent's ability to track and learn patterns and the overall environment. This leads to increased accuracy in predicting the Q-value of each action, in turn yielding better performance.

Examining performance for the agents trained in the environment with Random and Right ghosts, we see contrasting results. For the agent trained among Random ghosts, win rate decreases as the layer count increases, and both win rate and average score are maximized with a layer count of 3. For the agent trained among Right ghosts, both average score and win rate are maximized with a layer count of 4.

There are multiple explanations to these phenomena. One likely explanation is that increases in the number of layers leads to overfitting. Imagine an agent trained within the environment of Right ghosts. With more layers, the agent may learn that the ghosts move to the right when possible. Thus, when in the testing environment with the more hostile ghosts, they incorrectly assume the behavior of the ghosts to be less aggressive then they truly are. This leads to poor performance in the testing games. With less layers, perhaps the agents are only able to learn more broad generalizations, such as the way in which points are collected or to broadly evade ghosts, without learning their particular movement patterns.

Examining the results across axis of the testing environment, we see similarly interesting results. The MaxDirectional ghost environment, for every layer count, yields the worst overall performance during testing, while the Right ghost environment yields the best performance during testing. Overall performance increases notably as layer count increases.

4.3 Performance for 50,000 Training Games

We then test across various training environments using the layer count for the DQN that maximized performance for each agent. Thus, 5-layer DQNs were used for the agents trained in the environment with the Directional and MaxDirectional ghosts. For the agents trained in the environment with Right and Random ghosts, 4 and 3 layer DQNs, respectively, were used. Figures 5 and 6 show the average scores and win rates of the agent when exposed to varying environments for 50,000 training games and then tested for 1,000 games in each environment, by training environment and for the corresponding layer count in the DQN. Table 1 shows the average score and win rate for teach agent across the aggregation of the tests. Table 2 shows the aggregated agent performance in each testing condition.

Examining overall trends, we see that agent performance significantly increases with the increase of training games by an order of magnitude. We then consider the specific results, by training and testing conditions.

For the agent trained among Random ghosts, performance was maximized when tested among Right ghosts and minimized under the conditions with Directional and MaxDirectional Ghosts. This is a trend found across each of the other agents, as well. Overall, however, we see quite significant, positive performance in each testing condition. This implies that training the agent among Random ghosts is a rather effective training method, as it prepares the agent for a diversity of testing environments without overfitting the agent into learning one particular pattern for ghost movement.

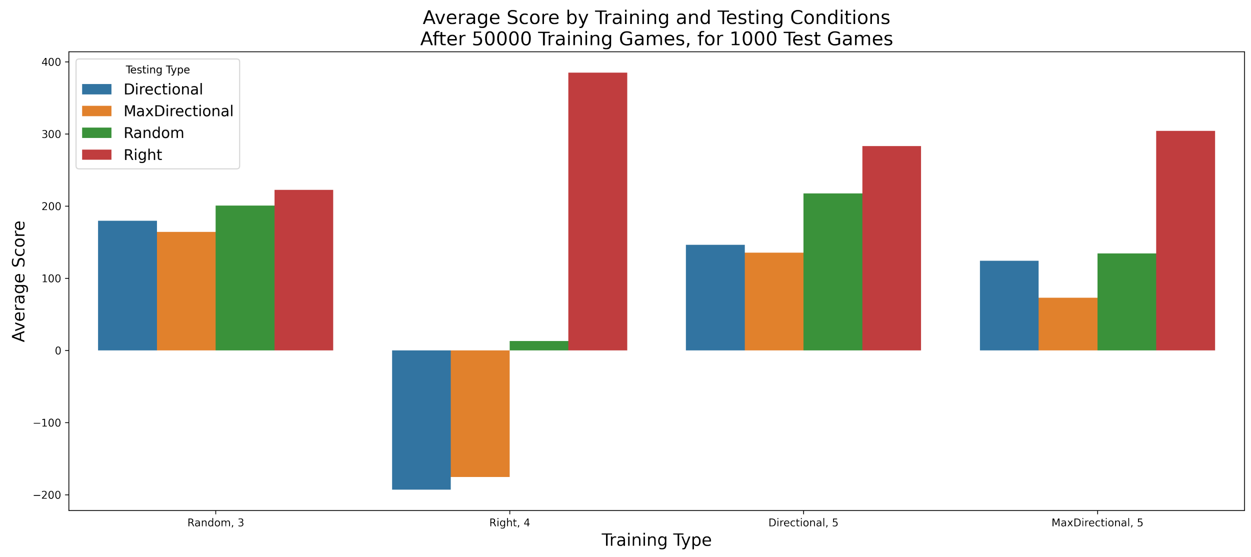

Figure 5: Average Score for Performance-Maximizing DQN Layer Count by Training and Testing Conditions

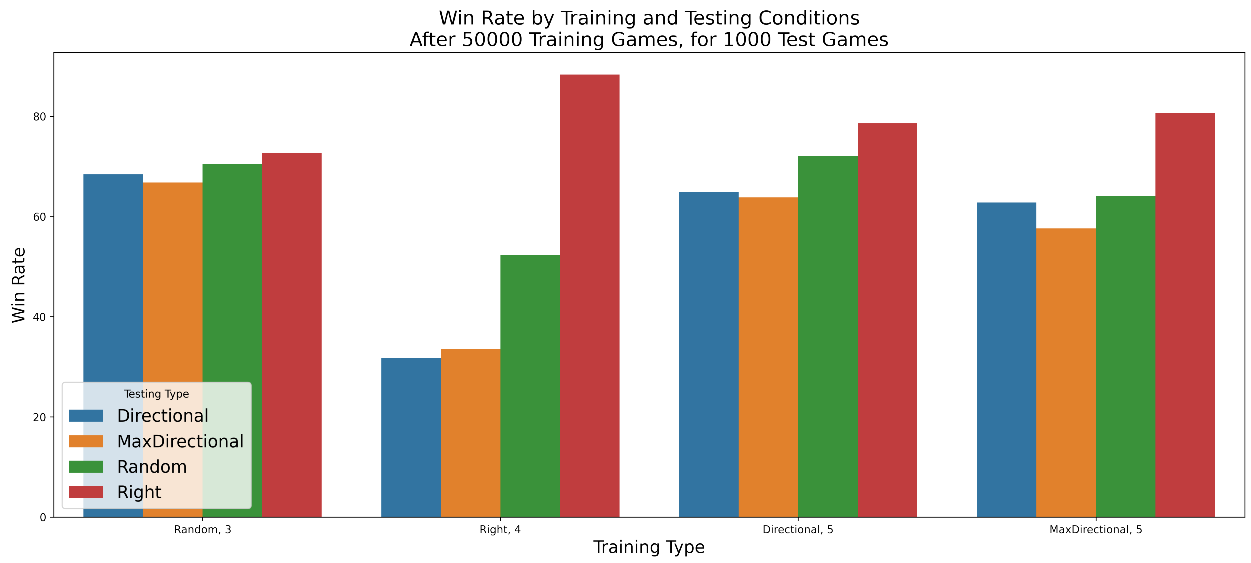

Figure 6: Win Rate for Performance-Maximizing DQN Layer Count by Training and Testing Conditions

Training Conditions	Layers	Average Score	Win Rate (%)
Directional	5	195.70175	69.850
MaxDirectional	5	159.04025	66.300
Random	3	191.91700	69.600
Right	4	7.53350	51.475

Table 1: Performance Across Training Conditions and DQN Layer Count

Testing Conditions	Average Score	Win Rate (%)
Directional	64.38775	56.975
MaxDirectional	49.46800	55.425
Random	141.50525	64.750
Right	298.83150	80.075

Table 2: Performance Across Testing Conditions

The agent trained among Right ghosts had, by far, the worst performance. This is in accordance with expectations. While other agents all had average scores of at least 150, the average score for this agent was less than 8. This is even given the fact that this agent in the environment with Right ghosts had the highest performance in a given test environment out of any other combination, by both average score and win rate. The likely explanation for these results is simple. Across the 50,000 training games, the agent is able to learn the right-moving behavior of the ghosts in the environment and, in the testing games, expects the ghosts to continue this behavior. Expecting the more aggressive, hostile hosts to behave as if they were the less aware, right-moving ghosts, they fail to evade the ghosts and, overall, perform suboptimaly.

The agent trained among Directional ghosts performed best by both average score and win rate. Though this was not the result hypothesized, there is a likely explanation. An agent trained among directional ghosts, who may still move randomly a still-significant 20% of the time, has experienced a broader array of ghost movements and overall outcomes. This is more evident when considering that this agent performed better than the agent trained among MaxDirectional ghosts in the environment with Random ghosts. The agent trained among Directional ghosts is better equipped to handle Random ghosts, as it has, itself, experienced some slight randomness in ghost variations. Incorporating slight randomness in ghost movement patterns prevents an agent from overfitting as significantly as some other agents do when learning ghost movement patterns. This allows the agent to focus on other, more important details crucial to success across testing in other environments.

The agent trained among MaxDirectional ghosts also performed well in each of the testing environments. However, we focus on one key aspect in discussing this agent's performance. It did not perform as well as well in the environment of its "native" MaxDirectional ghosts as the agents trained in the environment with Random and Right ghosts. The most likely explanation is that, across the 50,0000 training games and 5 layers of the DQN, the agent was able to rather effectively learn the movement patterns of the MaxDirectional ghost. However, despite its significance, this is not the most effective factor deserving of focus when learning to play the game. Likely, the other agents exposed to more randomness during training focus on other aspects, such as learning to attack ghosts when they are vulnerable or learning effective paths to quickly traverse the map in the time-sensitive game.

4.4 Demo Video

To provide visual insight into the results, a demo video is provided here (`https://youtu.be/x20gtH5MsAA`).

5 Implications & Further Work

There are significant implications to the results discovered. We particularly highlight the overall optimal performance of the agent trained among Directional ghosts. This result in particular likely shows the importance of training an agent in an environment that is close to the testing conditions as possible. The slight, yet significant, randomness of the Directional ghosts allowed the corresponding agent to react well in its"native" environment, as well as those with the Right, Random, and MaxDirectional ghosts. In real-world applications, we thus see the importance of focusing on training using environments (and datasets) that lead to focus on more generalized interactions and elements, rather than key details (such as key patterns of ghost behavior). Thus, in the self-driving vehicle example, we would train an agent across a multitude of street types and environments.

Considering the agent trained among Right ghosts, we see that an agent trained in an environment with an extreme lack of diversity of behavior will perform very suboptimally. Thus, while intuitive, one must be sure not to train an agent in an environment where a key feature has one specific value, especially if we expect that feature to change in testing. Otherwise, as seen, the results indicate quite poor performance.

Considering the agent trained among MaxDirectional ghosts, we see that agents trained to operate in the most hostile environment may not always perform the best overall, potentially even in their own environment. In the self-driving vehicle example, we note it may not be optimal to train an agent on the hardest, most traffic-ridden streets to learn city driving or the fastest, most crowded highways to learn highway driving. This is rather unintuitive at first and is quite notable, with countless other applications as well.

Future work could discuss the potential that an agent trained with a DQN of a given layer count may increase in efficiency after 50,000 training games, beyond the DQN with the specific layer count used. Furthermore, work could be done in another generalization of this experiment, perhaps in a situation such as training a self-driving agent to navigate various streets or training an agent to navigate various warehouses.

References

Dittadi, A., Träuble, F., Wüthrich, M., Widmaier, F., Gehler, P. V., Winther, O., Locatello, F., Bachem, O., Schölkopf, B., and Bauer, S. (2021). Representation learning for out-of-distribution generalization in reinforcement learning. *CoRR*, abs/2107.05686.

Koch, J., Langosco, L., Pfau, J., Le, J., and Sharkey, L. (2021). Objective robustness in deep reinforcement learning. *CoRR*, abs/2105.14111.

Padakandla, S. (2020). A survey of reinforcement learning algorithms for dynamically varying environments. *CoRR*, abs/2005.10619.

Padakandla, S., J., P. K., and Bhatnagar, S. (2019). Reinforcement learning in non-stationary environments. *CoRR*, abs/1905.03970.

Pinto, T. and Vale, Z. (2020). Contextual q-learning. *ECAI 2020*, Volume 325.

van der Ouderaa, T. (2016). Deep reinforcement learning in pac-man.