Can a Tetris Learner Create a Strategy to Minimize the Risk from Environment Randomness?

Kyle Miller

December 10, 2021

Abstract:

Every real-world environment contains some degree of risk and uncertainty. There is a need to create machine learners that are capable of creating strategies to reduce the risk they encounter from the unpredictable aspects of their environment. This project uses a neural network playing Tetris to determine what is necessary for a learner to begin to form risk-reduction strategies. Restrictions were applied to existing neural networks, and changes were made to the environment to determine what parameters lead the learner to create and utilize risk-reduction strategies. The study demonstrates that optimized state representations, along with imposed restrictions on the capabilities of the agent, can lead an agent to develop robust risk-reduction strategies instead of merely reacting to unpredicted aspects of the environment as they appear.

I. Introduction:

This project exists to determine what is necessary to get a reinforcement learning agent trained in a Tetris environment to develop strategies to reduce the risk that arises from randomness, as opposed to the purely reaction-based behaviors that generic Tetris learners tend to develop. Risk in Tetris arises from the fact that tetrominos fall in random orders, meaning a random string of tricky pieces could end an otherwise strong run. The problem in the Tetris environment can be observed in the way reinforcement learners behave when compared to the behaviors of world-record holding human players. Human players have deduced a powerful strategy to reduce the risk that comes from the fact that tetrominos fall in random orderings; leave one of the outer columns open so 4 rows can be cleared at once whenever an I-piece drops. However, in RL agents, this behavior is either absent or takes extraordinarily long to develop due to the local minima problem. The question then is what changes can be made to the agent, environment, or elsewhere to help the learner overcome a local minima and develop more robust risk-reduction strategies?

II. Methods:

The agent that was used for this project was a Deep Q Neural Network (DQN). A DQN was used as they are designed to plan for and prioritize future rewards over immediate rewards, which is especially important for this problem, where planning for the future is a large part of the desired end behavior. For the environment, it was decided to use a 2-dimensional array representation of the Tetris board. This was chosen over having the agent watch images of a "real" game of Tetris being played and learning from that, as the agent's ability to translate a series of static images into a game that it understands how to control is beyond the scope of this problem. Furthermore, the 2-dimensional array would be condensed into nine statistical values determined using Dellacherie's algorithm. The nine statistics represent aspects of the board such as number of holes, height of tallest column, etc. These values are then passed to the learner as the state. This was done so the agent would have a far simpler state to learn from, and so that the control experiment would closely match the behaviors and capabilities of state-of-the-art Tetris agents, which also heavily favor the use of Dellacherie's Algorithm for their state representations. With these in place, the control experiment would then be the DQN trained on Dellacherie's Algorithm over a period of 24 hours.

The next experiment was to test how the agent would perform when restrictions on its abilities were imposed. By default, the agent is capable of observing the entire board, the incoming piece, and all possible configurations of the board after placing the piece, all in a single frame. This gives the agent a massive advantage over human players and may be the reason why it is unable to develop any meaningful strategies – it may be simply too good at the game to need to develop proper plans the way a human does. The agent was therefore limited by lowering its reaction speed; first by 2x, then by 3x. It would now take the agent a non-insignificant amount of time to determine what the next best move is, which will result in it losing the game far quicker if it does not determine better strategies. Like the control experiment, these agents were allowed to train on Dellacherie's Algorithm for 24 hours.

The next experiment involved a change to the state representation rather than the agent. Instead of using Dellacherie's Algorithm to generate the statistical values fed into the DQN, a lesser-known algorithm known as EL-Tetris was used. This algorithm condenses the state into even fewer statistics, focusing on clearing rows rather than cells, and generally yields higher scores than Dellacherie's Algorithm. Likewise, the DQN was given 24 hours to train on the EL-Tetris environment. Two additional agents were also trained on the EL-Tetris environment, but with limited reaction speeds in the same way as the previous experiment.

III. Results:

Each agent was trained for a period of 24 hours and then evaluated.

Control – Unmodified Dellacherie's Algorithm

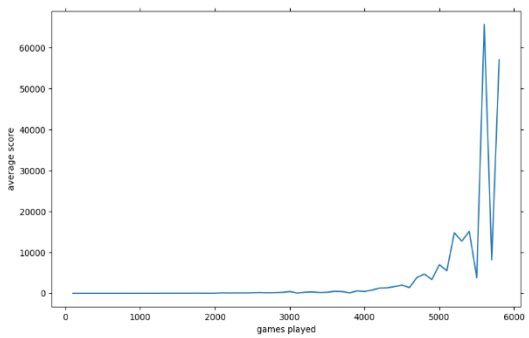

Fig 1: The learning curve of the control experiment over 24 hours of training.

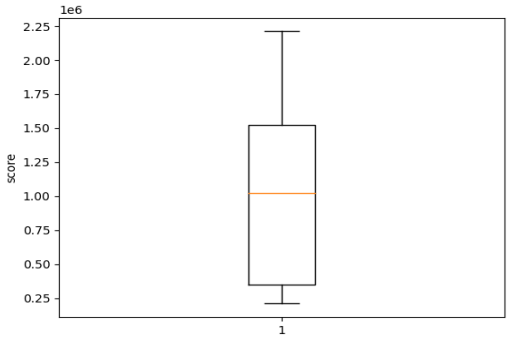

Fig 2: The scores of the control agent after training.

The control agent did not display behavior matching that of the optimal risk-reduction strategy. It was able to achieve an average score of 1,000,000, and a maximum score of 2,200,000.

Dellacherie's Algorithm – ½ Reaction Speeds

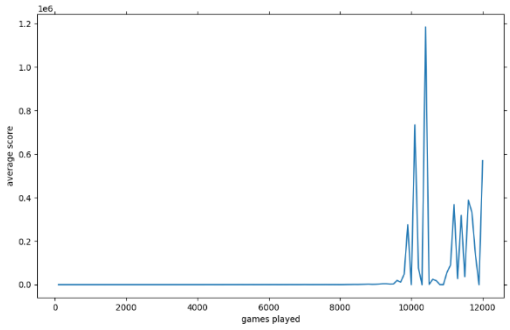

Fig 3: The learning curve of the ½ reaction speed agent over 24 hours of training. Notice the very sudden spike in performance– this reflects the agent learning the risk-reduction strategy.

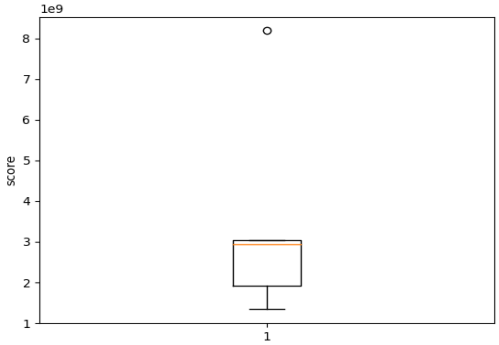

Fig 4: The scores of the ½ reaction agent after training.

The ½ reaction speed agent did display behaviors matching that of the optimal risk-reduction strategy. It was able to achieve a mean score of 3,000,000,000 and a max score of 8,200,000,000.

Dellacherie's Algorithm – 1/3 Reaction Speeds

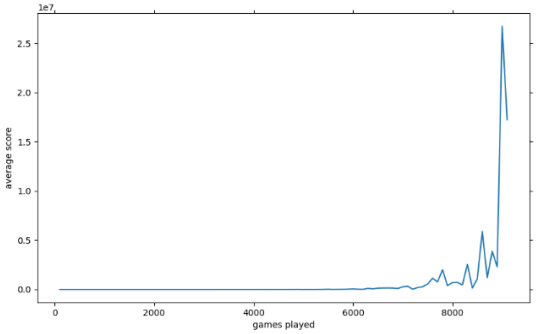

Fig 5: The learning curve of the 1/3 reaction speed agent after 24 hours of training. Notice the spike in reward happens several thousand games earlier than the ½ reaction speed agent.

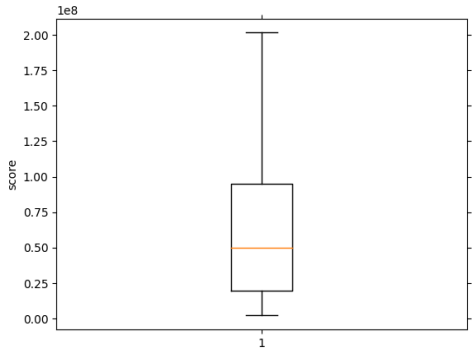

Fig 6: The scores of the 1/3 reaction speed agent after training.

The 1/3 reaction speed agent displayed behaviors matching that of the optimal risk-reduction strategy. It was able to achieve a mean score of 50,000,000 and a max score of 200,000,000.

EL-Tetris Algorithm

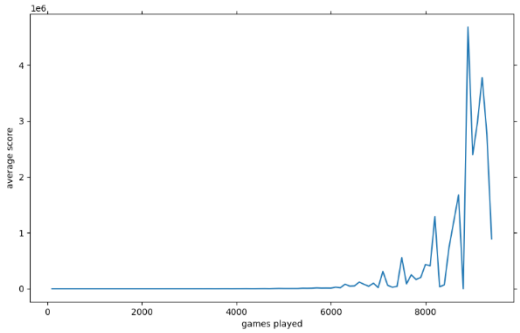

Fig 7: The learning curve of the EL-Tetris agent after 24 hours of training.

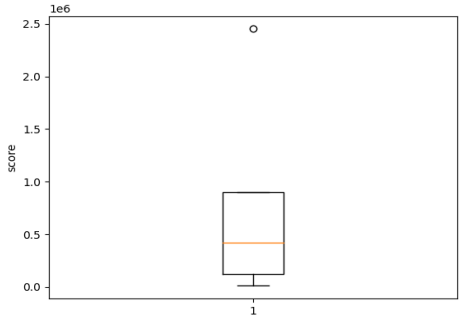

Fig 8: The scores of the EL-Tetris agent after training.

The EL-Tetris experiment displayed behavior matching the desired risk-reduction strategy after 24 hours of training. It was able to achieve a mean score of 500,000 and a max score of 2,400,000.

EL-Tetris Algorithm – ½ Reaction Speeds

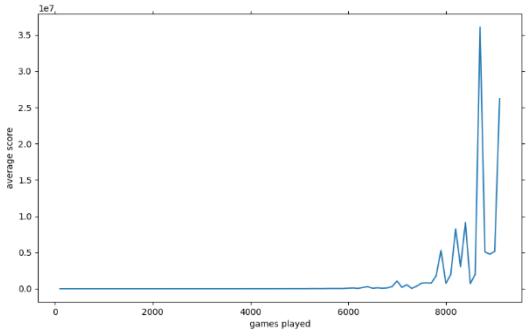

Fig 9: The learning curve of the ½ reaction speed EL-Tetris agent after 24 hours of training.

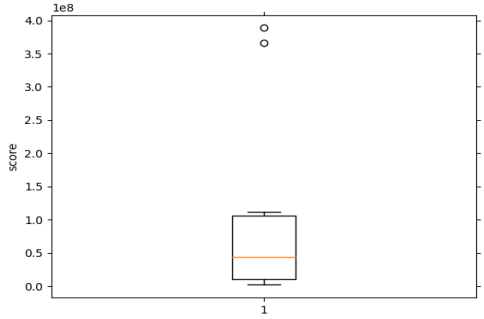

Fig 10: The scores of the ½ reaction speed EL-Tetris agent after training.

The EL-Tetris agent with ½ reaction speeds did not display behavior matching that of the optimal risk-reduction strategy. It was able to achieve a mean score of 50,000,000 and a max score of 400,000,000.

EL-Tetris Algorithm – 1/3 Reaction Speeds

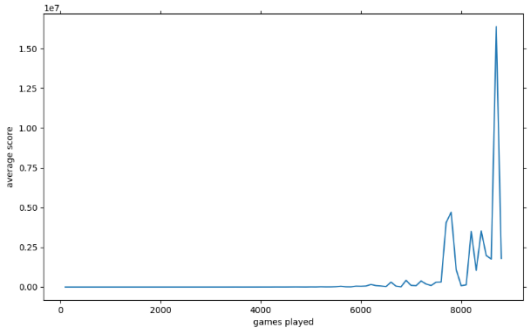

Fig 11: The learning curve of the 1/3 reaction speed EL-Tetris agent after 24 hours of training.

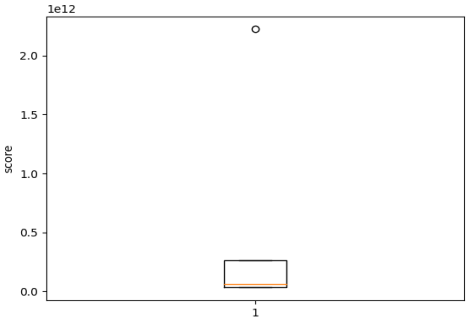

Fig 12: The scores of the 1/3 reaction speed EL-Tetris agent after training.

The EL-Tetris agent with 1/3 reaction speeds displayed behavior matching that of the optimal risk-reduction strategy. It was able to achieve a mean score of 150,000,000,000 and a max score of 2,200,000,000,000.

Comparison of Scores

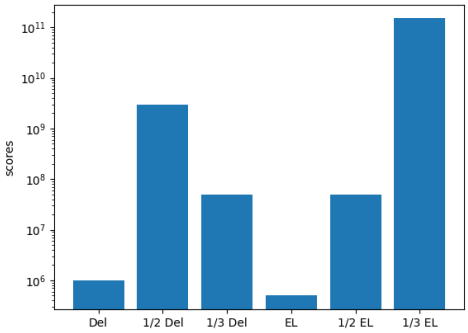

Fig 13: The mean scores of the agents

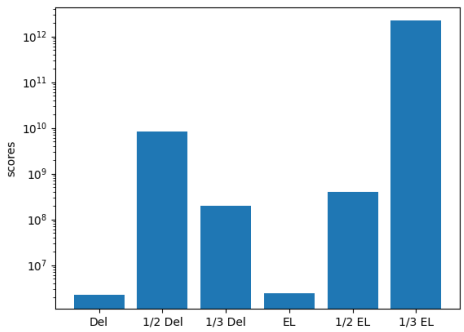

Fig 13: The max scores of the agents

IV. Summary:

The RL agent was able to overcome the local minima problem and develop an optimal risk-reduction strategy when optimizations were made the state, restrictions were imposed on the agent's capabilities, or both. However, the improvements yielded from these restrictions do not appear to behave predictably, at least within the tested domain. The 1/3 reaction speed agent performed worse than the ½ reaction speed agent for Dellacherie's algorithm, while the opposite held true for the EL-Tetris algorithm. In addition, the agent unlearned the risk-reduction strategy on the EL-Tetris environment when the ½ restriction was imposed. This implies that the specifics of what restrictions must be imposed on the agent to yield the desired results likely depends on a number of factors, most notably the state representation.

V. Conclusions:

It is possible to get a reinforcement learning agent on the Tetris environment to efficiently overcome a local minima problem and develop a more robust risk-reduction strategy through a combination of state optimizations and restrictions to the agent's capabilities. The techniques explored in this project may be applicable to more general problems in reinforcement learning, not just to the specific topic of risk-reduction in Tetris. It is possible that local minima problems may be overcome by using agent restrictions to "raise the bar" of what is an acceptable end strategy for the agent, forcing it to develop better strategies.

VI. Sources:

El-Ashi, Islam. "El-Tetris – an Improvement on Pierre Dellacherie's Algorithm." *Imake*, 1 June 2011, https://imake.ninja/el-tetris-an-improvement-on-pierre-dellacheries-algorithm/.

https://github.com/michiel-cox/Tetris-DQN