Creation of a Self Learning Tetris Agent Through the Application of Reinforcement Learning.

Tetris is a rather simple game, but humans are near incapable of reaching a point past 1000 lines. As the pieces continue falling and speeding up it becomes progressively more and more difficult to keep up. Thats where a self learning agent can be introduced. Through using the power of quick thinking along with trial and error the agent learns patterns and general drop placement that allow it to easily surpass human limits. In this project I have followed the steps of my predecessors in the reinforcement learning field to implement a tetris player beyond human capabilities.

Within this report I will discuss the problem, the methods I went through of fixing the problem, and what results I've received from pursuing this project.

Problem Faced

Tetris is a Nintendo video game created in 1984. It is initially an empty board of size 10 wide x 20 tall. The game is played as different shaped pieces are initialized at the top of the board and slowly fall down to the bottom. As they reach the bottom and stick it becomes a challenge to stack them appropriately so they don't reach the top of the board, the losing condition. When blocks are placed all the way across a row they are eliminated, giving the player a line as a well as a score for clearing that line. More score can be received for clearing more lines simultaneously. This creates a game where the challenge is either clearing the most lines, or getting the most score.

The problem faced by this game is that as time goes on the blocks fall faster and faster making it nearly impossible in the past to pass a certain number of lines cleared for human players.

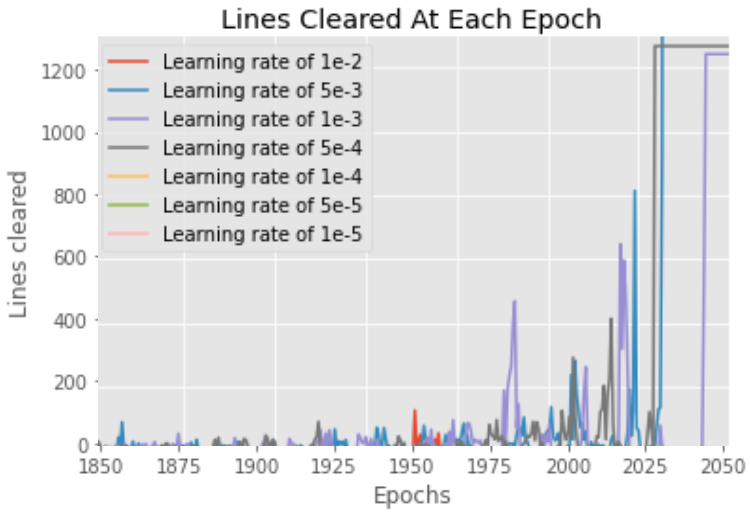
So given the tetris board and the random set of blocks given in a tetris game I wanted to create an agent that can find patterns and block placements such that we can clear over 1,000 lines. Thus solving the game in a way that a human could never hope to achieve.

Methods

For this project I applied a Q-learning algorithm to determine the what is the most ideal way of transitioning to another state. Since the piece order of tetris is random it is not always common to have the exact same board state as a past event the agent needs to approximate the value of a board state and push towards ones with more value. This also allows us to allow the agent to interact with tetris in a different manner. With the way implemented the agent will look at all possible locations that a piece may be placed and determine the value of that placement. This makes so teaching the agent to move is negligible as instead they can just choose whiver location is providing the most value.

Results

Results are found by viewing how many lines the tetris agent achieves over training simulations. For example on the first run of the agent achieves very few lines cleared. By the 100th simulation it has started to clear a few line or two and by the 2000th simulation we have achieved the desired 1000 clines cleared and discontinue the tetris agent from running. Below you can view the results of the training at different learning rates.



The agent was trained to view the speed of breaking line 1000 with the three quickest learning rates taken from the initial graph.This reduces the randomness possibility of randomness by finding the average of the three learning rates. Thus we view that the average rate of achieving 1000 lines for the 5e-3 learning rate is 1990. The average for 1e-3 was 2077.33 and the average for 5e-4 was 2282.66.

Summary

Summarizing the project a tetris agent was created and trained to clear as many lines as possible. Given the nature of tetris if a agent can clear lines infinitely then it can achieve an infinite score. After it was able to learn it was tested to find the learning rate that resulted in the fastest 1000 lines cleared averagely. This being the learning rate of 5e-3 with an average of 1990 epochs before it was able to break 1000 lines cleared.

Conclusions

Tetris is a game that requires quick thinking and equally as quick ability to move. By implementing a self learning agent we can achieve near instantaneous thinking and movement that a human could never hope to achieve. A score and line limit that a human can set can end up inconsequential to a long trained agent. 1000 lines cleared is almost an impossibility for a human and yet within 2000 practices a tetris agent with the right learning rate can easily clear that benchmark.

Citations

Applying Artificial Intelligence to Nintendo Tetris. Applying artificial intelligence to Nintendo Tetris. (n.d.). Retrieved December 1, 2022, from https://meatfighter.com/nintendotetrisai/

Bakibayev, T. (2020, December 30). *How to write a tetris bot in python*. Medium. Retrieved December 1, 2022, from https://levelup.gitconnected.com/tetris-ai-in-python-bd194d6326ae

Kauten, C. (2019). *Gym-tetris*. PyPI. Retrieved December 1, 2022, from https://pypi.org/project/gym-tetris/

L, R. (2021, October 5). *Reinforcement learning on Tetris*. Medium. Retrieved December 1, 2022, from https://medium.com/mlearning-ai/reinforcement-learning-on-tetris-707f75716c37

Melax, S. (n.d.). *Reinforcement Learning Tetris Example*. Reinforcement learning Tetris example. Retrieved December 1, 2022, from https://melax.github.io/tetris/tetris.html

Stevens, M., & Pradhan, S. (n.d.). *Playing Tetris with deep reinforcement learning*. Playing Tetris with Deep Reinforcement Learning. Retrieved December 2, 2022, from http://cs231n.stanford.edu/reports/2016/pdfs/121_Report.pdf

Code Sources

Nguyen, V. (2020, March 30). *Uvipen/tetris-deep-Q-learning-pytorch: Deep Q-learning for playing Tetris Game*. GitHub. Retrieved December 1, 2022, from https://github.com/uvipen/Tetris-deep-Q-learning-pytorch

Faria, N. (2021, July 19). *Nuno-Faria/Tetris-ai: A Deep Reinforcement Learning Bot that plays tetris*. GitHub. Retrieved December 1, 2022, from https://github.com/nuno-faria/tetris-ai

My implementation

https://github.com/dec3ptive/Tetris