AI learning Tetris using genetic algorithm

František Špaček

spacefr1@fit.cvut.cz

7.1.2024

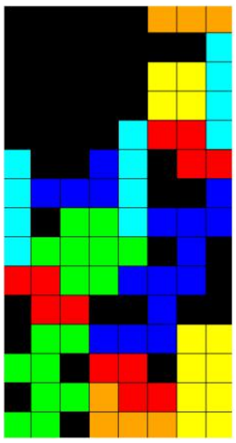
1) Introduction

The objective of this semestral work is to teach a neural network how to play Tetris using a genetic algorithm. This requires a creation of a basic game environment. Variables in this environment can be modified via a GUI. To help with this task, the app has to have a way to visualize the AI progress and a way to save networks for later use.

Many such projects exist, but they don't let the AI play like a real player. Instead it is being fed precalculated positions and the neural network only chooses one of them. This work will instead allow the AI to move the piece at any time with the gradually falling down, thus playing like a living person.

The result of this work should ideally be a neural network that is able to play Tetris like without making mistakes and potentially surviving forever.

2) Input and output



The play area has default size 15 * 8 which is less than normal, but it should

The input for the neural network will be the block type, rotation and position. Instead of telling the AI about every block in the grid, it will only be given a list of the highest blocks in each column to minimize the amount of data. It will not see holes under some pieces, however it should be able to play in such a way that these never even appear. The network has four output nodes for moving left, right, rotating and dropping the block.

3) Possible challenges

The main problem of teaching the neural network is that there is no good way to grade it's individual decisions. Instead it has to play the whole game. This makes it difficult to use backpropagation.

Instead a genetic algorithm will be used which will play hundreds of games at the same time and using the results for the next generation. This will most likely lead to very slow learning as every change will be a shot into the dark.

For that reason, the first attempts will let the AI play with only one type of block to make it easier a quicker. Slowly adding block in next attempts.

4) Evaluation criteria

The goal is to only learn how to play. It is not required to have the AI complete multiple lines at once to get additional points. The score of the neural network is calculated at each step as the y value of the falling block with bonus for each block placed. This will push the neural network to place as many block as low as possible, in the best case completing lines and staying on the ground. During the testing, it became apparent that the AI has trouble with rotating the block so it often decides to not do it at all. To push it towards the desired goal, it will receive a tiny reward for each rotation. To prevent the creation of holes (covered places where no block can be placed), points will be deducted when a hole appears.

5) Selection

At the begging the selection was done by taking the better half and killing the rest. I found this to not be a good approach as it lead to the algorithm being stuck in a dead-end.

The approach I settled with is a combination of elitist and roulette wheel selection. This keeps a few of the best performing individuals unchanged and fills the rest with random clones (with slight mutations) where there is a bigger chance the better performing networks will be selected, but gives a chance to all the other ones and increases the diversity. To allow for enough changes to occur, a large population is required, in this case it was from one hundred to over a thousand for more complex tasks.

6) Results

The neural network has successfully learned how to play with a single piece. This took different amounts of time depending on the shape. It was able to play perfectly with O and I pieces in just two generations. Other shapes took a bit longer. The size of the network was two hidden layers with twenty neurons each.

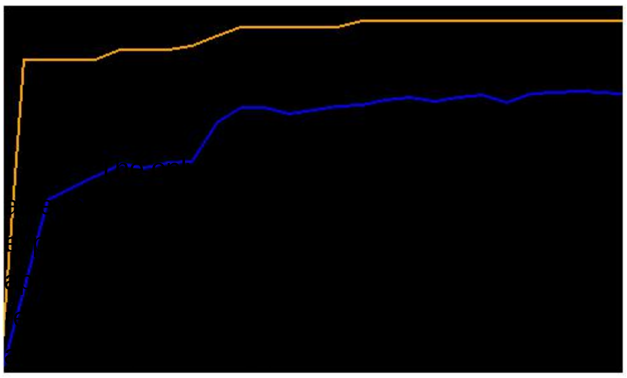
Complications arose with multiple pieces. A bigger network was required which unfortunately dramatically increased the time required. It is possible to teach it how to play with two pieces, the most I managed was four which took a very long time.

The AI was never able to learn how to play well with all pieces. Only trying to fit as many as possible and completing a few lines at best.

7) Conclusion

The project was a mixed success. The neural networks and the genetic algorithm work without trouble. The history of best and average results is displayed in a graph along with additional info about current generation.

It is possible to modify the environment using a graphical user interface and it’s possible to save and load networks.



unpractical when compared to other possible solutions, like letting the AI choose from a few preselected options. To make the real time playing neural network possible, something more sophisticated than simple genetic algorithm should be used. That is a direction for possible future improvements.

8) **Sources**

Python AI: How to Build a Neural Network & Make Predictions [online]. [cit. 2023-12-7]. Dostupné z: https://realpython.com/python-ai-neural-network/

Tetris [online]. 2021 [cit. 2023-11-20]. Dostupné z: https://github.com/rajatdiptabiswas/tetris-pygame

How To Create a Neural Network In Python: With And Without Keras [online]. 2022 [cit. 2024-01-02]. Dostupné z: https://www.activestate.com/resources/quick-reads/how-to-create-a-neural-network-in-python-with-and-without-keras/

Pygame tutorials [online]. 2021 [cit. 2024-01-04]. Dostupné z: https://github.com/russs123/pygame_tutorials

Stack overflow [online]. 2017 [cit. 2024-01-07]. Dostupné z: https://stackoverflow.com/questions/46390231

Beating the world record in Tetris (GB) with genetics algorithm [online]. ANH BUI, Duc. 2020 [cit. 2024-01-07]. Dostupné z: https://towardsdatascience.com/beating-the-world-record-in-tetris-gb-with-genetics-algorithm-6c0b2f5ace9b