Multiplayer Tetris Using Deep Q Learning and Transfer Learning

Suraj Vathsa, Viacheslav Inderiakin, Ian Conceicao

Abstract

Tetris 99 is a multiplayer battle-royal game extending classic Tetris gameplay. In this work, we create the game environment, implement training algorithms and optimize reinforcement learning agents to play Tetris 99 at the human level. In particular, we compare the performance of three Neural Network architecture classes: a Fully Connected Net, a Fully Connected Net with artificially constructed features and a Convolutional Neural Network. We also compare three learning algorithms: Deep Q-learning, Deep Q-learning with fixed targets and Double Deep Q-learning. In addition, we study how transfer learning affects resulting performance. We show that by using preprocessed features and keeping model size moderate it is possible to achieve high performance in both single-player and multiplayer tasks. We also show that transfer learning helps greatly enhance model's performance and guide training.

Introduction

Tetris 99 is a game where 99 players compete to be the last player standing. Each of the players controls a standard 20×10 board where they can sequentially place incoming pieces. Completion of certain conditions, for example, clearing 4 rows at once, not only increases score, but also gives an opportunity to attack other players by overloading their boards with incomplete lines called garbage rows, or simply garbage. Our team chose this game over other RL testbeds because it combines four features frequently present in real-life RL problems.

- First, it has an enormous state space of more than 2^20 states for the single-player mode and an exponentially higher state space in the multiplayer scenario.
- Second, the environment is not deterministic.
- Third, it requires the agent to be aware of the other players and learn a strategy to survive in a competition.
- Finally, Tetris 99 holds high similarity to its single-player version, which makes the usage of transfer learning possible. Simultaneously, to the best of our knowledge, this game has not been studied before in a reinforcement learning context.

Methodology

We explore three distinct approaches for Neural Network design.

- In the first scenario, we pass the raw representation of the players' boards through several fully connected layers with ReLU activation functions.
- The second approach implies the usage of a set of artificially constructed features including: board height, number of cleared lines, "bumpiness" and "number of holes". The artificial features are then passed through several fully connected layers.
- Finally, we try to pass the raw board state through a combination of convolutional and fully connected layers, to compensate for the large state space complexity while simultaneously retaining high final performance.

We attempted Deep Q Learning, Deep Q Learning with fixed targets, and Double Deep Q Learning with these three architectures and recorded performance results.

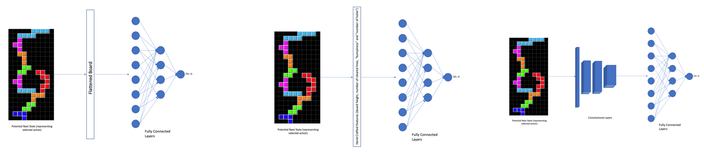


Fig2. Various approaches used to train vanilla Tetris agent.

Based on results obtained from training vanilla Tetris agents, we observed that fully connected layers with hand crafted features worked best. We used this as an inspiration to design the multiplayer Tetris 99 agents. Since in the game, each player has a view of every other player's board, feed opponent board features as well to the fully-connected network using an architecture shown below.

Key features of this architecture include:

- Features of interest of the player's board
- Features of interest of all the opponent player's board
- Equal weight for both opponents' board features and player's board features using latent representation of both inputs

Based on our experiments in single-player (vanilla) Tetris, we attempted to train models for playing Tetris99. Similar to the network architectures, the algorithms used for reinforcement learning were adopted from vanilla Tetris. Since Deep Q Learning with fixed targets performed best, we used this algorithms for training the (multiplayer) Tetris99 agent.

We restricted the number of players to two players given the limited time and resources available for training. Given the multiplayer nature of Tetris99, we sought to understand how the Deep Q Network that performed best in single-player Tetris and a modification to the Deep Q Network, inspired by this architecture, behaved when made to compete with an opponent skilled at vanilla Tetris at various levels.

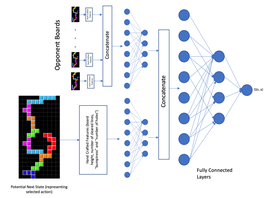


Fig 3. Neural Network architecture used for multiplayer agents

We also sought to understand the effects of transfer learning and whether transferring weights learned by an agent playing vanilla Tetris would help boost performance or training speed. To carry out the experiments we trained two different architectures in three scenarios:

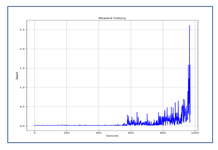
1. both the agent and its opponent start learning from scratch, and opponent's weights are regularly updated using agent's weights.
2. the multiplayer agent initialized with random weights plays against the best single-player agent, which weights were transferred from single player scenario.
3. both the agent and its opponent are initialized using the weights of the most performant agent from single player scenario; opponent's weights are regularly updated using agent's weights

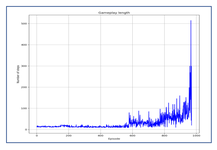
Results

Initially an attempt was made to design the T99 environment with raw button presses. However, after testing against a variety of RL algorithms and NN architectures, it was found that raw button presses require too long time to be learned on machines we have at our disposal. This is why we created a simplified environment that supports up to 2 players and uses snapshots of the boards after a certain number of steps as actions.

Training was monitored using reward accumulated in each episode, the total number of timesteps the episode lasted and the total number of lines that were cleared by the agent. Further, each combination of architecture and algorithm was evaluated for 50 epochs with exploration turned off and the cumulative rewards, average steps per episode, and average number of lines cleared per episode were calculated.

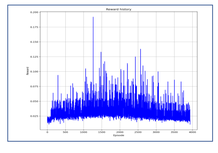
Vanilla Tetris with fully connected layers as Deep Q Network and preprocessed features fed as input. Deep Q Learning with fixed target was used for training.

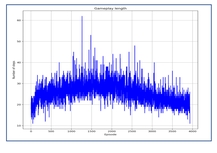




	$\mathbb{E}[\cdot]$	$\sigma[\cdot]$
Survival time	∞	N/A
Total Reward	∞	N/A
Lines Cleared	∞	N/A

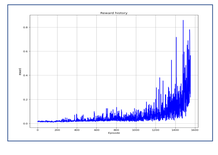
Vanilla Tetris with Fully connected layers as Deep Q Network and raw flattened board fed as input. Deep Q Learning with fixed target was used for training.

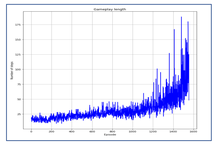




	$\mathbb{E}[\cdot]$	$\sigma[\cdot]$
Survival time	44.26	98.39
Total Reward	0.25	0.78
Lines Cleared	12.04	37.92

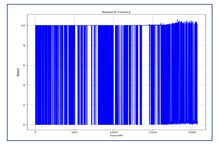
Vanilla Tetris with CNN based Deep Q Network with the raw board as input board. Deep Q Learning with fixed target was used for training.

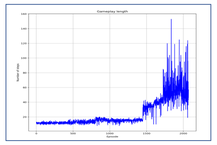




	$\mathbb{E}[\cdot]$	$\sigma[\cdot]$
Survival time	459.98	310.04
Total Reward	2.33	2.79
Lines Cleared	170.72	123.44

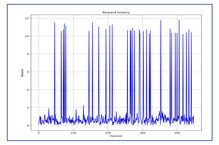
Two player T99 with agent and opponent trained from scratch

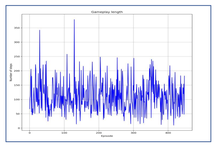




	$\mathbb{E}[\cdot]$	$\sigma[\cdot]$
Win Probability	0.76	
Episode Length	124.22	86.48
Lines Sent	6.24	3.44

Two Player T99 with agent trained starting with weights transferred from single player agent against an opponent with weights transferred from a single player agent. The opponent's weights are updated at regular intervals with new weights learned by the agent.





	$\mathbb{E}[\cdot]$	$\sigma[\cdot]$
Win Probability	0.92	
Episode Length	176.35	109.28
Lines Sent	18.15	8.89

Conclusion

During the course of this project, we trained RL agents that can play classic single-player Tetris and compete in multiplayer Tetris 99 at human level. We compared the performance of several architectures and training algorithms on both single-player and multiplayer testbeds, and observed that increase in feature complexity strongly increases training time and degrades final performance. However, given a highly representative features and an architecture with sufficient capacity, an agent can be trained to play almost perfectly mimicking human behavior. We also compared how transfer learning affects training in multiplayer scenario. We found that transferring knowledge of game mechanics from single-player scenario to multiplayer is highly beneficial for training the agent in the given number of steps. In the future, we are planning to study ways of transferring knowledge of obtained RL agents for environments that use raw button presses as inputs. We also hope to extend these experiments to T99 with 99 players and observe if the behaviors observed in a two player setting is the same as that observed in a ninety nine player setting.

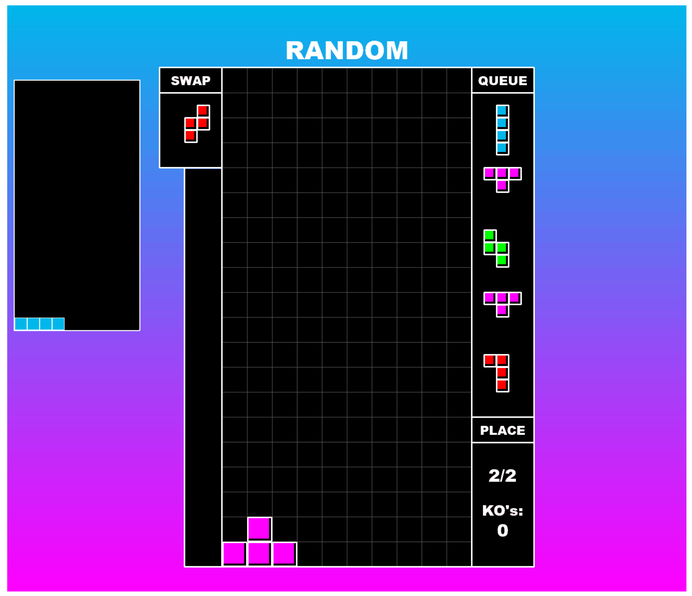


Fig 1. Sample T99 Gameplay