Tetris Battle – A New Environment for Single mode and Double Mode Game

Yi-Lin Sung
Graduate Institute of Communication Engineering
National Taiwan University
r06942076@ntu.edu.tw

Abstract

We introduce a new environment for simulating the once popular game on Facebook, so called "Tetris Battle", which can be viewed as an advanced version of Tetris. There are two modes in Tetris Battle. The first one is single mode, and the goal is to attain the scores as high as possible under the time limitation. The other is double mode, and both players aim to interrupt the other one by sending garbage lines to the opponent until one of the players cannot add more blocks to the board. We believe the proposed environment helpful for developing novel algorithms about reinforcement learning, self-playing and imitation learning, especially when the computational resources are not enough to train in complex environments, such as Go and StarCraft.

1 Introduction

Game playing is an active research area in both computer vision and control system. There are several advantages of studying game playing: (1) it is often accompanied with a close and relative simple environment (2) it is no harm to real world (3) there are complex behaviors such as planning when playing games. One could also found that many reinforcement learning (RL) techniques are often verified in games first ([1, 2]). Due to these reasons, developing more challenge environments is beneficial to both the research and practical field.

OpenAI gym [3] is the library containing comprehensive environments from robotics to games. It offers a platform to compare RL algorithms. Even if there are lots of single mode games, however, multi-agents games are still limited. And games with multi-player can be helpful in studying self-play algorithms. Therefore, this motivates us to develop a multi-agents games (at least two-agents).

AlphaGo and its variants ([4], [5]) defeat Go experts stunning the world that the machine can solve such a high dimension game. However, training a agent from scratch needs 3 days with 4 TPUs for AlphaGo zero, which is the most efficient version in variants of AlphaGo. It might take 18.5 days to train a model with a single 2018Ti, it is time consuming for the labs with limit resources. Note that StarCraft is a more complex enviroment than Go, and [6] indicates that the training takes 44 days with 12 TPUs. Therefore, a environment with moderate complexity will be advantageous for RL researches.

Combining the above motivations, we introduce a new environment, "Tetris Battle", which is the once popular game on Facebook. There are both single mode and the double mode, so it can not only used in developing self training algorithms, but also it can be used to validate the algorithms once used in single player games such as Atari games. Tetris Battle is a variant of Tetris. The board is gradually filled up as pieces of different shapes, called Tetriminos (Fig. 1). When the rows are full, they will be eliminate and the player get points. There are two modes in our implementation, which are single mode and double mode. In single mode, the player is asked to clear as more lines as

Accepted at the Workshop on Deep Reinforcement Learning at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

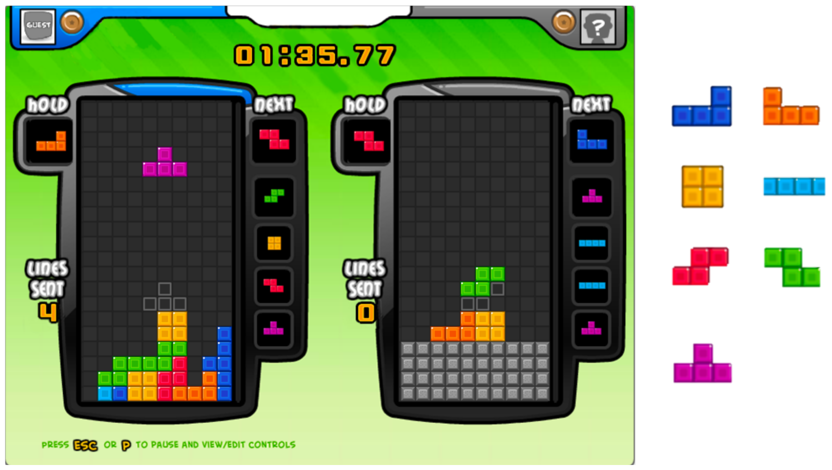

Figure 1: Illustration of Tetris Battle. Left: The screenshot of the game. And the bottom 4 gray lines on the right player board are garbage lines. Right: The shape of seven Tetriminos.

possible, while there are some additional bonuses for combos, tetris and tspin (they will be introduced later). Those bonuses are added to let agent learn how to do long-term planning. In double mode, both players aim to beat each other by sending garbage lines to the opponent's board. The garbage lines cannot be eliminated, so it will make the game harder. While one player's board has no more spaces for next Tetromino, his opponent is going to get a KO, and his opponent's garbage lines will be cleared. In addition, the player with 3 KOs wins the game.

Note that there are some works trying to master Tetris, such as [7, 8, 9, 10]. In [7], they utilize approximated policy iteration, which is implemented by dynamic programming , during training. Nevertheless, dynamic programming is intractable when the complexities of the states and actions become large. In [9, 10], these approaches all use additional processed features related to the game board, such as the aggregated heights and the number of the holes of current board. However, in our Tetris Battle, we use the screenshots as the game's states, and this kind of settings make the game harder to solve. The purpose of [8] is similar as ours. However, their methods all apply on traditional Tetris, whose goal is to survive given a sequence of Tetriminos. On the other side, in single mode of Tetris Battle, the goal is to reach high scores under time limitation. And the rules of Tetris Battle encourage the agent to get the long-term rewards while traditional Tetris lack of this property. Therefore, the objectives of the two games are different. Moreover, the double mode of our game will be a new challenge for the RL algorithms.

In this paper, we also explore the performances of the two popular algorithms, Advantage Actor Critic (A2C) [1] and Proximal Policy Optimization (PPO) [2], on single mode Tetris Battle under certain conditions. We find that the complexity of the game is too high when all 7 kinds of blocks are used, so we try to reduce the complexity of it by only using a subset of the blocks. The details are shown in Sec. 3. In this work, we only focus on dealing with single mode game and we leave the further studies on double mode one as future works.

2 Tetris Battle

2.1 Environment

A major contribution of our work is the environment set-up for Tetris Battle. Since Tetris Battle is no longer available on Facebook and the source codes are not released. Therefore, we re-implement the game and it is a highly restored version of original one[1], such as Fig. 1. Next, we wrap this game into an OpenAI gym-like environment [3] *env*. This will make existing algorithms built on OpenAI-gym easily apply to our environment with only minimal modifications. The primary function used to run the environment are:

```
# set up the environemt
# or use env = TetrisDoubleEnv() to create double mode game.
```

```
env = TetrisSingleEnv()
ob = env.reset() # reset the game
ob, reward, done, info = env.step(action) # conduct the action and update game
```

There are two kinds of observation of our environment: raw screenshots and the board information. Most of the algorithms based on OpenAI gym using the raw screenshots as inputs. However, in our environment, the size of the screenshot is 600×800, and it might be memory consuming and result in hard training. Therefore, we also provide the other kind of state, which are the direct information of boards, including the states of the board (there are two boards of two players, so the size is $2 \times 20 \times 10$), the following Tetriminos, the time information, etc. The size of this kind of state is 20×34 (we concatenate of the features together).

In Tetris Battle, there are eight actions. They are *idle*, *down*, *left*, *right*, *clockwise rotation*, *counter-clockwise rotation*, *drop* and *hold*. Some explanation of *hold* are in Sec. 2.2. The possible board configurations are 2^{200} (2^{400} for double players). All permutation of the holded Tetrimino and next 5 coming Tetrimino are $7 \times \binom{7}{5}$. The total possible scores are roughly from 0 to 200. Therefore, the complexity of the single mode game is about $2^{200} \times 7 \times \binom{7}{5} \times 200 \approx 2^{215}$ (2^{430} for double mode game). If we use the screenshots as the states, the complexity are even higher, but the latent complexity are still 2^{215}. Note that the state complexity of Go is 2^{361}, so Tetris Battle is a challenge environment (however, the actions space of GO is much larger than Tetris Battle). Note that we haven't include the continuous features such as remaining time. [11] states that even the whole sequence of Tetriminos are known, maximizing the cleared lines is still a NP-complete problem.

2.2 Rules

Tetris Battle shares most rules with traditional Tetris [12]. However, there are some different features between them:

1. The time limit of Tetris Battle is 2 minutes, while which is absent in Tetris.
2. In Tetris Battle, player can foresee the next 5 Tetriminos.
3. There is an additional action, *hold*. The player uses *hold* to store current Tetrimino to the memory and draws a new Tetrimino. If there is already a Tetrimino in memory, then switch the current one with it. Note that one can only use *hold* 1 time before one block falls to the ground.
4. The score in Tetris Battle is called "*line sent*". The player is going to get $n - 1$ sent when clearing n lines simultaneously. Therefore, eliminating only one line gets no point.
5. The player can get the additional sents by consecutive eliminating lines, and this behavior is called "combo". The player can get bonus 1 / 2 / 3 / 4 sents by 1 $\sim$ 2 / 3 $\sim$ 4 / 5 $\sim$ 6 / > 7 combos.
6. There are two special moves in Tetris Battle. The first one is T-spin. It is rotating the T-shaped Tetrimino to fill an otherwise unreachable slot at the very last moment. The player only clear 2 lines when using T-spin, but he can get 4 line sents. The other one called "Tetrix"[2]. When the player use I-shaped Tetrimino to clear 4 lines simultaneously, he can get 4 line sents (not 3 line sents).
7. There is a special flag called "back to back". If the player use T-spin or Tetrix in this turn, he will get the "back to back" flag. When the next turn he also uses T-spin or Tetrix, he can get 2 bonus sents. The flag will disappear after a normal elimination.
8. In double mode, the player can use "garbage lines" to interfere the opponent. When the player get n line sents, he will send n garbage lines, which cannot be cleared, to the other's board. The garbage lines will be created after the opponent's current Tetrimino falling to the ground. If he get m line sents by falling that Tetrimino, the number of garbage lines will be reduce to $n - m$.
9. In double mode, when the player's board has no space for the next Tetrimino, the garbage lines will be cleared, and his opponent will get a *KO*. They player with 3 KOs wins the battle. In single mode, once the Tetrimino has no space to put, the game is over.

[1]The enviroments are available in `https://github.com/louis2889184/TetrisBattle`

3 Training an Agent Using Reinforcement Learning

3.1 Brief review A2C and PPO

Deep reinforcement learning (DRL) uses neural networks and reinforcement learning principles to create an efficient and generalized algorithms. For DRL, the agent learns the policy π_θ and value function V_ϕ, where θ and ϕ are both parametrized by neural networks. The policy function decides the probability of each action given the current state while the value function evaluates the scores of the current state. During training, the agent receives observation s_t at each time step t, selecting an action at with probability $\pi_\theta(a_t|s_t)$, and then receives a reward r_t from the game environment. The objective of the agent is to maxmize the expected reward $\mathbb{E}_\pi(R_t)$, where $R_t = \sum_{i=0}^{n} \gamma^t r_{t+i+1}$ and γ is the discount factor.

A2C is an actor-critic method. It update its policy network's parameters via policy gradient $\nabla_\theta log(\pi_\theta(a|s))A(s_t)$, where $A(s_t, \phi) = R_t - V(s_t, \phi)$ is the advantage function. On the other side, value network is updated by the gradient $\nabla_\phi(r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t))^2$ by given the current action a_t.

The computational complexity of policy gradient method is too high for real tasks since it need to keep interacting with the environment. PPO is an off-policy version of actor-critic algorithm. The advantage of off-policy algorithm is that the model can train on the old policy multiple times and eases the need of interacting to the environments. However, we need to ensure that the old policy is close to the real policy, so that the approximation could be precised enough. Therefore, the gradient of policy network become $\nabla_{\theta'} log(\pi_{\theta'}(a|s))A(s_t)\frac{\pi_\theta(a|s)}{\pi_{\theta'}(a|s)} + \beta KL(\theta, \theta')$, where the $KL(\theta, \theta')$ is to ensure the distributions of two policies are similar.

3.2 Experiments

We only explore the single mode Tetris Battle in this paper, since it is already hard for training. We leave the double mode for future works. Moreover, In our experiments, we found that the training is unsuccessful when using all 7 Tetriminos. Therefore, in order to reducing the complexity of the environment, we only use a subset of Tetriminos. We also explore the two types of observation in our experiments.

In A2C and PPO, we training the models for total 700000 steps, and update the parameters every 32 steps. β is set to 0.01 for PPO.

Our reward function is design as $r =$ line cleared $+$ line sent. If we only use line sent as reward, then the agent is hard to get the reward, and the parameter has no chance to improve. By adding the number of cleared lines to the reward function, the reward's sparsity of the game is reduce, and the training become possible.

Table 1: The received rewards by using A2C and PPO under different subsets of Tetriminos. We report the highest value during training.

Methods	$\{I, O\}$	$\{I, O, L\}$	$\{L\}$
A2C	0.0	0.3	0.1
PPO	39.2	14.1	9.1

In Table 1, 2 and 3, A2C is failed to train under all the settings. This might because that A2C is sensitive to the hyperparameters and we don't make effort to tune them. Agent using PPO is successfully trained. It is surprised that our agent cannot master the game with only L block, even the complexity of this setting is less than which of the others. In the setting of only L block, the agent have to learn how to rotate, or it will lose quickly. We suspect that rotation is a hard behavior, and we will delve into and solve the problem in the future.

Table 2 and 3 are the results of our experiments of two kinds of observation. It's hard to believe that the agent using high dimensional raw screenshots reach the better results. This result might due to

[2] Its original name is Tetris. However, to reduce the confusion with the name of the game, I use Tetrix in this paper instead.

Table 2: The rewards of two methods on single mode Tetris Battle. We report the highest value during training. The used Tetriminos are I and O.

Methods	raw screenshot	board information
A2C	0.0	0.3
PPO	39.2	16.3

Table 3: The line sents of two methods on single mode Tetris Battle. We report the highest value during training. The used Tetriminos are I and O.

Methods	raw screenshot	board information
A2C	0.0	0.1
PPO	23.2	6.7

two reasons. The first is that we use the deeper network for the raw screenshots inputs. The second is that the color information is helpful during training, e.g. it can be used to distinguish the Tetriminos. And we just use 0 and 0.5 and 1 to indicate that each spaces on the board are empty, occupyied by current Tetrimino and occupied. We will study this issue in the future.

4 Conclusion and Future Direction

We introduce a new environment called Tetris Battle, which is support for single mode and double mode. We hope that Tetris Battle can be used in most labs with different computational resources. We hope that Tetris Battle could be the assistance for developing RL algorithms.

The future work is listing as follows:

- Try more algorithms to play Tetris Battle, and successfully train a agent in both single and double modes.
- Explore how the different settings influence the RL algorithms.
- Implement more features for the game.

References

[1] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

[4] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[5] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lynette R. Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Hui zhen Fan, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[6] Oriol Vinyals, Igor Babuschkin, Wojciech Marian Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, Duck Hwan Choi, Richard W. Powell, Timo Ewalds, Petko Georgiev,

Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[7] Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of tetris. *J. Mach. Learn. Res.*, 16:1629–1676, 2015.

[8] Matt Stevens. Playing tetris with deep reinforcement learning. 2016.

[9] Donald Carr. Applying reinforcement learning to tetris. 2005.

[10] Yiyuan Lee. The (near) perfect bot. `https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/`. 2013.

[11] Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell. Tetris is hard, even to approximate. In *COCOON*, 2003.

[12] Simón Algorta and Özgür Simsek. The game of tetris in machine learning. *ArXiv*, abs/1905.01652, 2019.