Week -1 (18.10 – 24.10)

19/10

- Start-up, watched presentation, setup basic structure for project work folder.

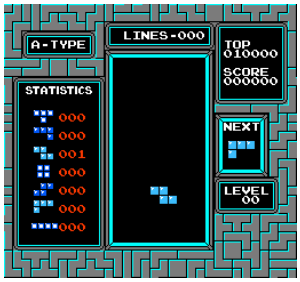
Week 0 (25.10 – 31.10)

- Busy due to assignments and obligatory coursework in other courses.

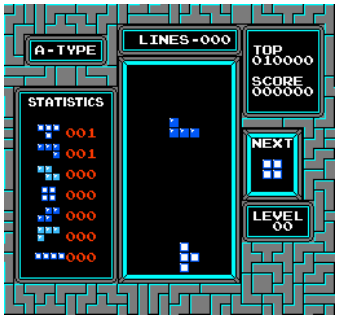
Week 1 (01.11 – 07.11)

Thursday 04/11

- Chose task Reinforcement Learning.
- Need to decide on environment. Dabbling in the idea of creating own, but this could take a while to implement so settling for a pre-made one for the time being.
    - https://github.com/Kautenja/gym-tetris
- Trying to prepare dependencies etc., encountering issues with Visual C++.
- Got the environment running.



- Experimenting with the environment to figure out the variables. Using the "SIMPLE_MOVEMENT" action space since it has all the inputs we require for Tetris.



- For future reference:

0 – NOOP	No Operation, agent idles
1 – A	Agent rotates piece clockwise
2 – B	Agent rotates piece counterclockwise
3 – right	Agent moves piece to the right
4 – left	Agent moves piece to the left
5 – down	Agent soft drops piece

Friday 05/11

- Decided on trying to implement the AI using DQN and referenced the 8th exercise. Quickly ran into a wall and decided to start looking into how to save and load trained models before doing anything else, as this will probably prove to be rather important later, since Tetris is significantly more complex than the CartPole environment. I therefore expect the training to take quite some time.
- Noticed some flawed logic in the CartPole DQN model, specifically with the exploration rate, where the agent would never explore anything. This leads to the agent only being able to evolve if it gets "lucky" in the first few episodes.
- Implemented a fix that lets the agent learn more consistently, although it does not always retain what it learns.

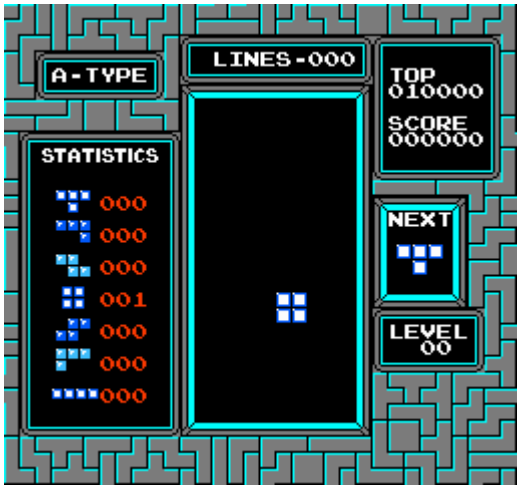
Week 2 (08.11 – 14.11)

Wednesday 10/11

- Noticed that the previous fix seems to have broken the model somewhat, in the sense that it is no longer able to continue from where it was after saving and loading.
- Researched some others' implementations of Tetris AI, seems that most people create their own environments.
- Attempts at fitting the model to the chosen environment, encountering difficulties as there isn't really any documentation, which makes it hard to figure out what exactly each component is.

Friday 12/11

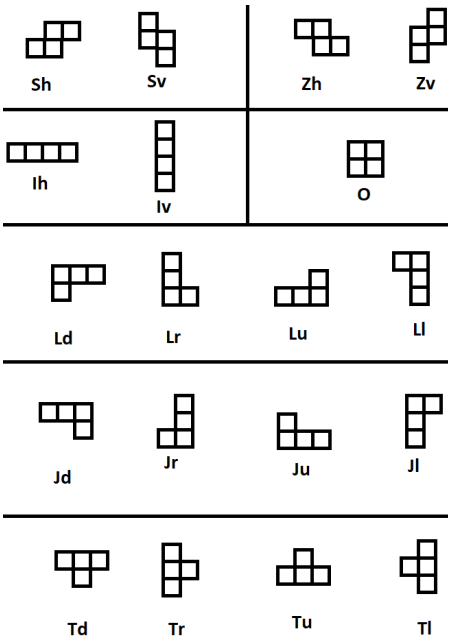
- Resumed attempts at trying to fit model. Trying to figure out what exactly the state returns etc.
- Attempting to use the "info" that the environment returns instead, as there is some documentation on what these are. A simple solution lets the agent play with some degree of intelligence, but I could not get it to learn properly.

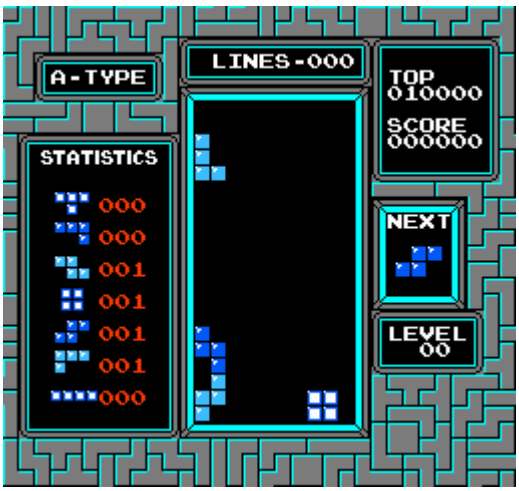


Week 3 (15.11 – 21.11)

Tuesday 16/11

- Trying to fit the model again. Finally realized that the 240 x 256 x 3 state that the environment returns is each individual pixel of the 240 x 256 window, as well as the RGB color values.
- Had a meeting with Jonathan, who recommended trying a different environment since the one I had chosen wasn't the greatest to work with. Introduced an environment developed by last year's students of TDAT3025, https://github.com/JLMadsen/TetrisAI, and suggested using this one and focusing on some interesting way of implementation instead of the developing of an environment in itself.
- Decided to play around with the Kautenja environment for a little longer just to see where I'm at before I move on. Left the agent to play for a large amount of time and recorded its progress, but when I came back it had crashed on episode 24. After reviewing the footage, I found that the crash was caused by the agents first line clear, which is very unfortunate as this was a chance to see if the agent would evolve after finally finding a way to increase its score.
- Previously used a simplified way of referring to current piece, investigated how the env referred to the different pieces' rotations and found them as can be seen on the right. Letting the agent know this additional information did not seem to help an awful lot.
- Experimented with letting the agent get rewarded simply by staying alive, but this seemed to make the agent rather hyper-active. It's likely that the agent understood this reward as "press as many buttons as possible" instead of "build as low as possible to stay alive", since the reward gained from mashing outweighed the

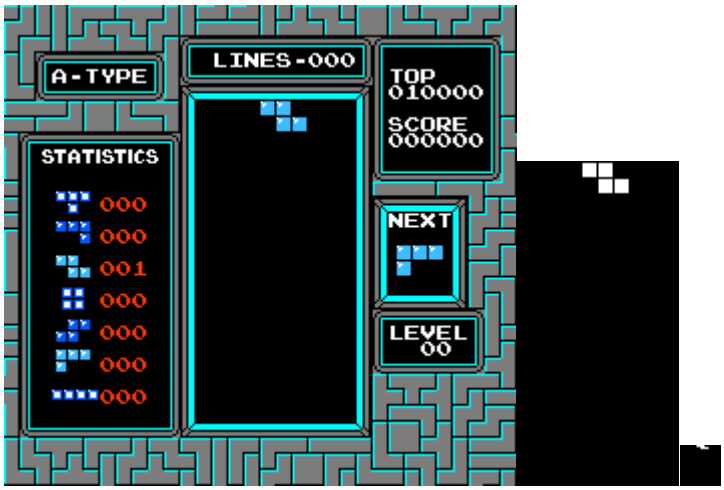




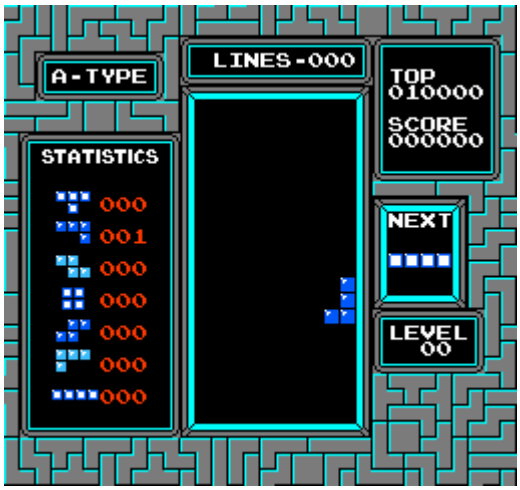
- Concluded that my general understanding of Deep and Reinforcement Learning is somewhat lacking, so I'm going to spend some time getting a grip on these topics before moving on to the new environment.

Wednesday 17/11

- My stubborn nature will not let me move on past the current environment before I've achieved something. Spending the day researching and trying to implement a way for the agent to read the pixel state of the game.
- Found a useful article on converting the state of an Atari game and playing them with DQN. After reading and following its instructions I've reached a model that runs, all that is left is to let it run for a while and see if it learns.
- Realized the model was over-cropping the observations, so it lost information at the top and bottom of the Tetris matrix. Here's what the image processing process currently looks like:

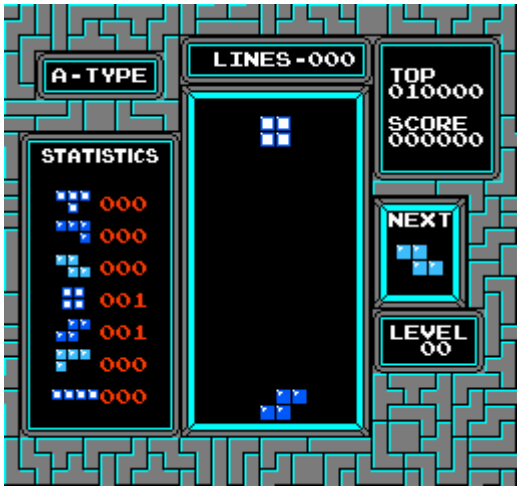


- After tweaking the model for roughly 6-8 hours and not seeing any real improvements I've decided to leave it to train for a while before coming back.
- The optimizer was practically unreachable, not very shocking that the agent wasn't learning.
- Model seems to like building upwards, maybe because it's the only place anything happens? Will try implementing a bonus for staying alive to see if it outweighs it, since the only current reward is clearing lines (something the agent is unfortunately unable to do).
- As with the previous model, award the agent an extra point for each frame it stays alive doesn't seem to have any real impact on its performance, other than making it press as many buttons as possible.
- After observing the agent play for a while it seems to connect the gain with certain actions, as can be seen by it repeatedly smashing the tetronimos into the side.



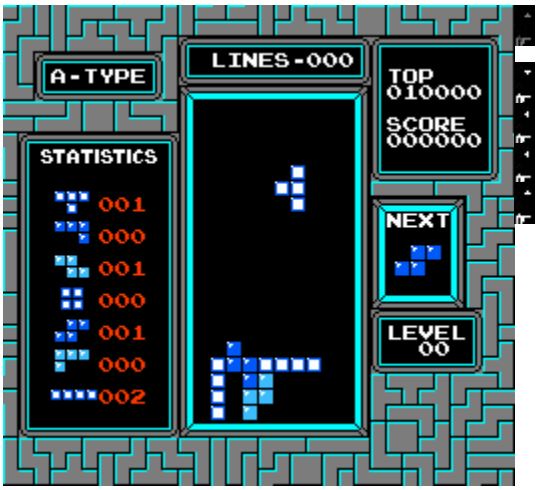
Thursday 18/11

- After leaving the agent to train for a long period of time it seems the agent does appear to develop some tendencies, although not the ones we want. Instead of trying to clear line it seems to try to end the game as quickly as possible, and ends many episodes within 1000 steps, the lowest thus far sitting at 428. Typical episodes in the earlier episodes last anywhere from 5000 to 10000 steps.
- Stopped the agent at episode 612 without seeing too much improvement. Loading the model for evaluation looks like so :
- No real progress today, will wait reconvene with Jonathan about how to proceed before committing to anything big. Might start looking into heuristics and genetic algorithms.

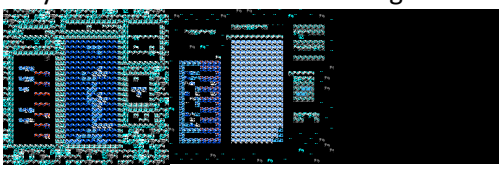


Friday 19/11

- Met with Jonathan, confirmed that the project is going alright but that I will indeed have difficulty seeing progress with the original Tetris environment due to its complexity.
- Concluded that after some finishing touches on the DQN model, it might be a good idea to move on and start looking at heuristics and genetic algorithms.
- Revamped the image processing and to accept a rectangle image instead of our standardized square. Also noticed how the different pieces were being perceived differently due to their gradients, which should no longer be a problem. Going to experiment with syncing how many frames are skipped such that every action will have its own frame when buffered, and then giving only 2 layers to the model. Might give a better of idea of which action does what? Will let the agent train freely for a while and see if there's been any noticeable change.
- For reference, processing currently functions as follows, with last actions being (from latest to earliest): 1, 0, 1, 5, 2, 5, 3



encountering some graphical errors with the environment. After letting it run for a while, it may become one of the following:

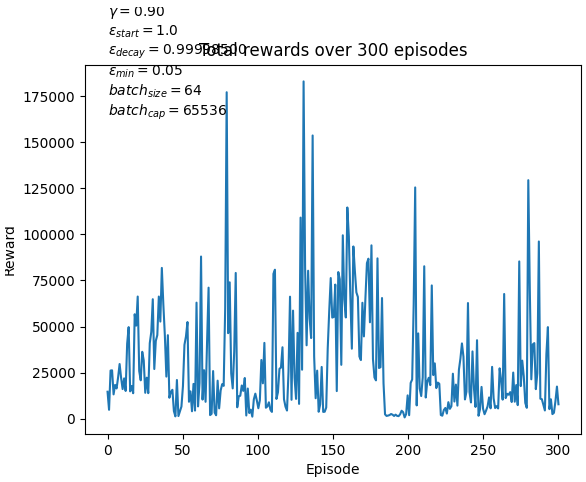


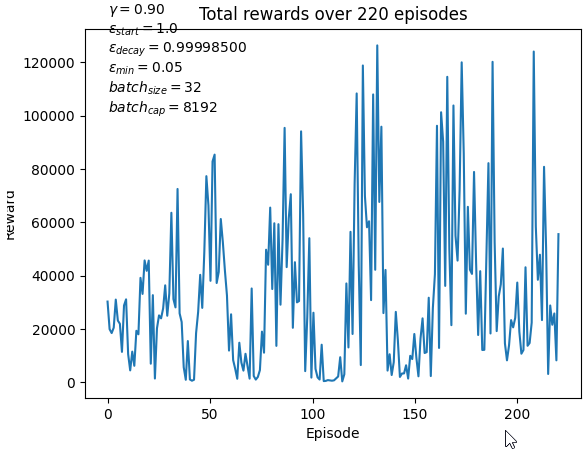
Week 4 (22.11 – 28.11)

Assignment turn-in 26.11!

Monday 22/11

- The current DQN model has the disadvantage of taking a long time to train, and with the graphical bugs and time-per-episode on the environment I'm using it will likely be difficult to get it trained to a point where it will perform well. Nonetheless, I believe it has proved to have some intelligence based on the way it has acted.
- Will start researching and looking into how I can implement heuristics while attempting to let it train in the background to gather statistics.
- Two training sessions with different batch sizes and memory capacities. (Rewarded for staying alive).





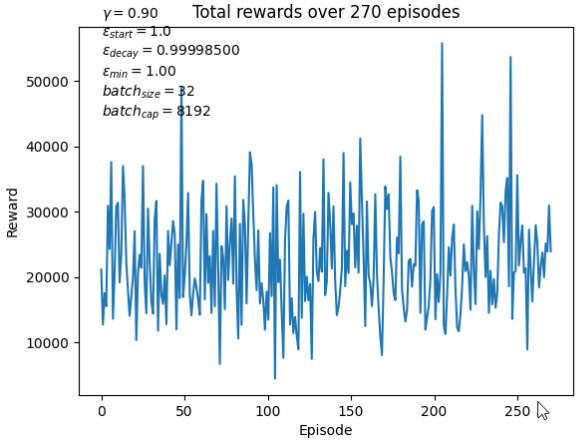


Figure 1 and 2 use the same hyperparameters except for the ones mentioned above.

Figure 3 has a minimum epsilon of 1.0, meaning that every step was explorational and thus random. Including this graph for comparison.

Tuesday 23/11

- Think I've got a reasonable hang on the theory behind using heuristics and genetic algorithms. Will start implementing features needed for these, starting with those that will analyze the game state and return heuristics.

Wednesday 24/11

- D

Thursday 25/11

- D

Friday 26/11

- Turn-in!