**Playing Snake with Deep Reinforcement Learning**

Introduction:

Reinforcement learning (RL) is a subset of machine learning that involves an agent to take an action in a deterministic environment (state) and has been a popular algorithm used in playing various games to outperform human. In reinforcement learning, the agent will aim to achieve to select the best action in order to maximize the rewards by employing trials and errors in a given current state [ ]. As the learning progress, the agent will find the best action based on the Q-table consists of a map of action-state pairs to rewards after accumulating short-term rewards to a long-term reward. However, as the state taken from the game gets complicated as agent would always encounter new behaviours, the Q-table is no longer feasible for infinite spaces. This has pose a challenge on traditional reinforcement learning. A deep reinforcement learning (DRL) is introduced to tackle this issue as the policy decision is evaluated through the deep neural network instead of a table.

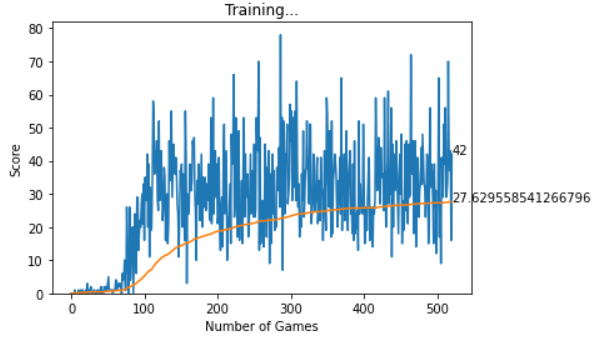
Intro:

* Define the problem?
* How people / related work on the problem

Methodology:

* Block Diagram
* The Q-values are updated according to Bellman equation

Results / Experiment:



|  |
| --- |
| **3.**  1. Food direction from the head:  Up, Down, Left and Right  3. Immediate danger from the snake head:  Front, Left and Right.  2. Last action taken:  Up, Down, Left and Right  **2**  **1.** |

|  |
| --- |
| If not last transition, Target max Q =  Otherwise, Target max Q =  Backpropagation, *Loss*  Sample a transition  No  Yes  store (s, r, s’, a)  Replay Buffer, *D*  Sample minibatch of 32 transitions  If game over  Target  maxa, j Q (sj, aj)  clone  Predicted  maxa, j Q (sj, aj)  maxa', j Q (s'j, a'j)  No, state insert  If random < epsilon, ε  maxa Q (s, a)  action, a  Yes, generate random number  state, s  array of Boolean |

d