WID3009 Mini Project: DQN-Based Dino Run

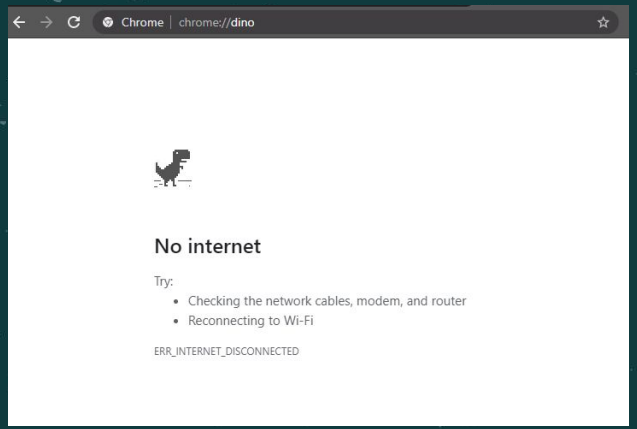
Group Members

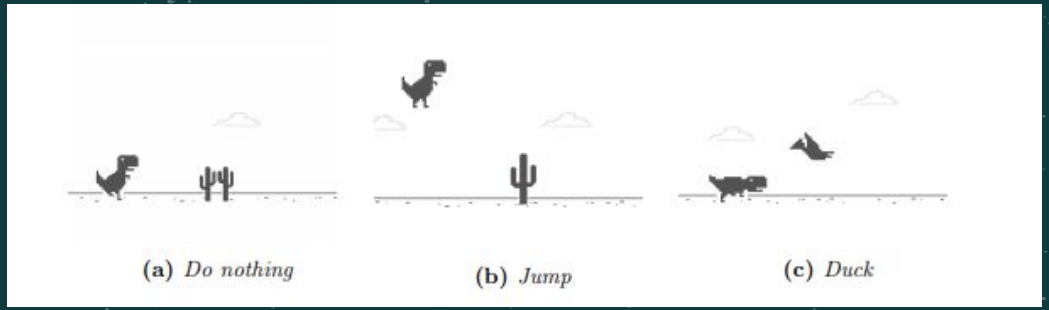
1. Lim Jia Qi 17134267/1
2. Wong Hao Shan 17122789/1
3. Chong Sin Mei 17103500/1
4. Cheah Jo Yen 17059391/1

Introduction

What is Dino Run?

- A game that appears in Google Chrome in offline model
- In the game, the dinosaur can be controlled by space/up and down arrow keys to jump and duck to avoid the obstacles in the game.
- Can be accessed with link: chrome://dino/





Introduction

What is DQN?

- Deep Q Network

- based on off-policy reinforcement learning model Q Learning and uses a Convolutional Neural Network to learn the game-specific presentations.

- Enhancements to the original Q-learning algorithm
  - Experience replay buffer
  - Freeze the target network

Objectives

- To develop a Deep Q-Networks (DQN) agent to control the dinosaur to maximise the scores of the game
- To evaluate analyze the performance of the model developed according to q-value, rewards and scores

Related Work

AI for Chrome Offline Dinosaur Game

- Deep Q-learning, online learning Multi Layered Perceptron (MLP) and rule-based decision-making (Human Optimization) for learning to control the game agent in T-Rex.

Deep Q-Network Atari Pong Game

- Implementation of the Deep Q-learning algorithm on the Atari Pong environment.

Requirements and Technical Implementations

OpenAI Gym Environment

- Observations, Actions and Rewards
  - The observation is a RGB numpy array with shape of (150, 600, 3).
  - The available actions are 0: do nothing and 1: jump.
  - A positive reward 0.01 is given when the dinosaur is alive; No penalty is given for early jump
  - A negative penalty -1.0 is given when the dinosaur hits an obstacle and die.

Selenium

- to make an interface between our python code and the browser-JavaScript code because the game is in JavaScript

OpenCV

- Pre-process the image to reduce their dimensionality

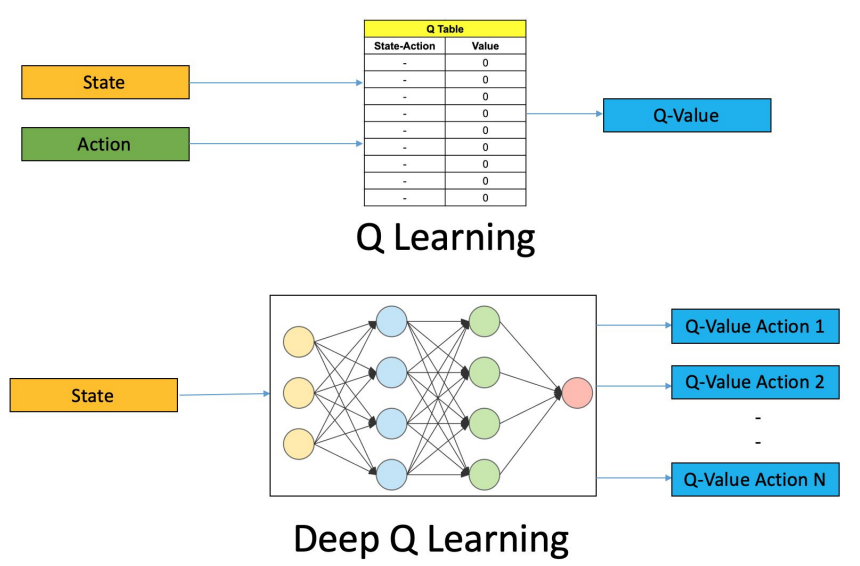
Pytorch

- a python based library to provide flexibility as a deep learning development platform

Methodology

Deep Q-Network(DQN)

- Deep Q-Learning is a combination of the Q-Learning algorithm and deep neural networks.
- In deep Q-learning, a neural network is used to approximate the Q-value function.
- Helps the agent figure out the next action to perform in order to maximise the reward



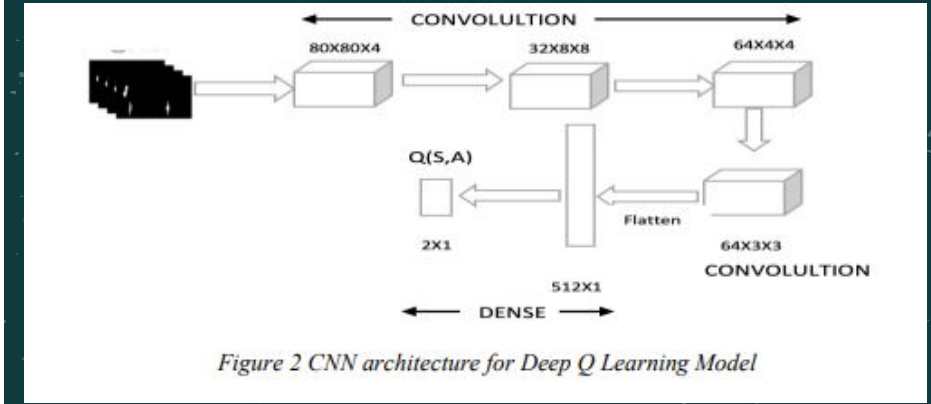
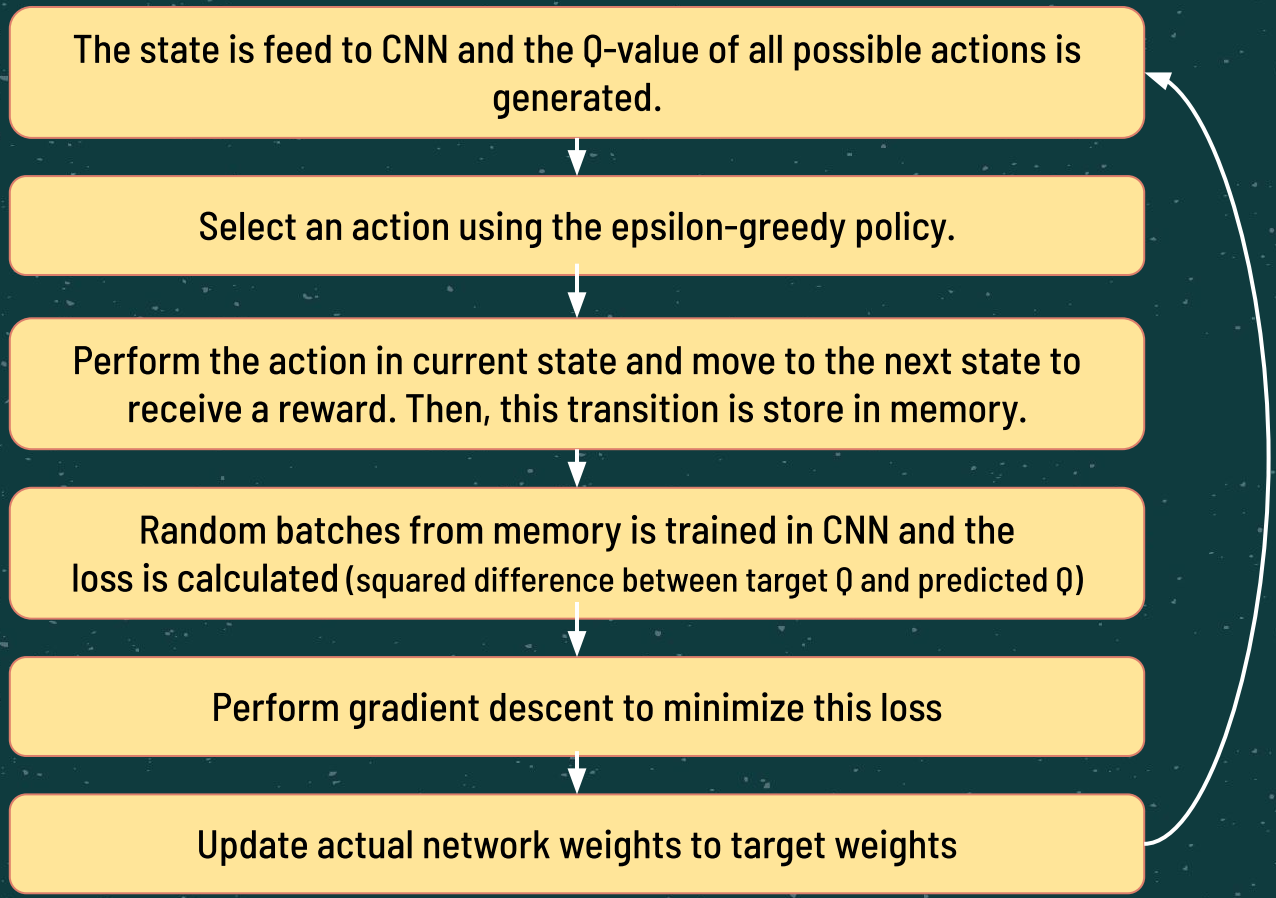


Figure 2 CNN architecture for Deep Q Learning Model

Deep Q-Network(DQN)



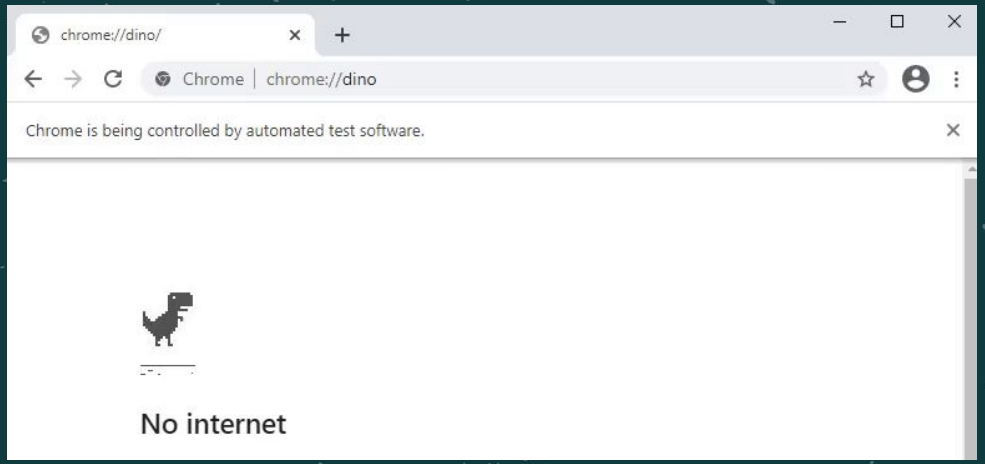
Batch Training

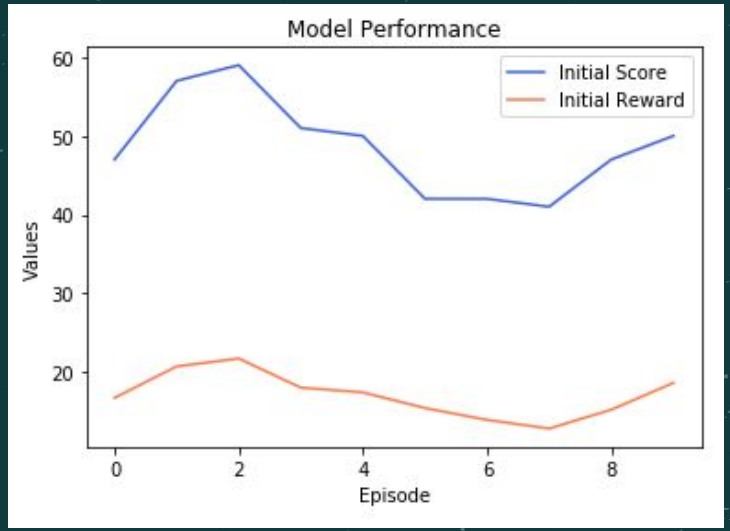
- If we train the CNN at every single frame, the training process will be very slow because the state at time t is **highly correlated** with state at time t+1 and the gradient descent after consecutive steps will cause erratic updates.
- To make the training process faster and more stable, batch training method is used.
- During training, we maintain a memory with capacity 30,000 and save the observations at each frame.
- A batch of data from the memory is then chosen randomly to train our CNN and this experience replay strategy makes the training much more efficient.

Training Parameters

- To make the agent explore more states and policies, we implemented the **epsilon-greedy algorithm** to do random action
- The choice of initial and the total steps of exploring are essential because:
    1. The agent can not control itself and take any action when the T-Rex has already jumped into sky. *(even a small $\mathcal{E}$ will make the agent die very fast)*
    2. Difficulty of the game increases gradually *(the agent can not explore any more during the high speed mode of game when the T-Rex is kept alive for a long time)*
- The value of epsilon, $\mathcal{E}$ decreases linearly over time
    - The initial epsilon, $\mathcal{E}$ is set at 0.1 which is then decay for every step until it reaches 0.0001
    - The total steps of exploring is set at 100,000

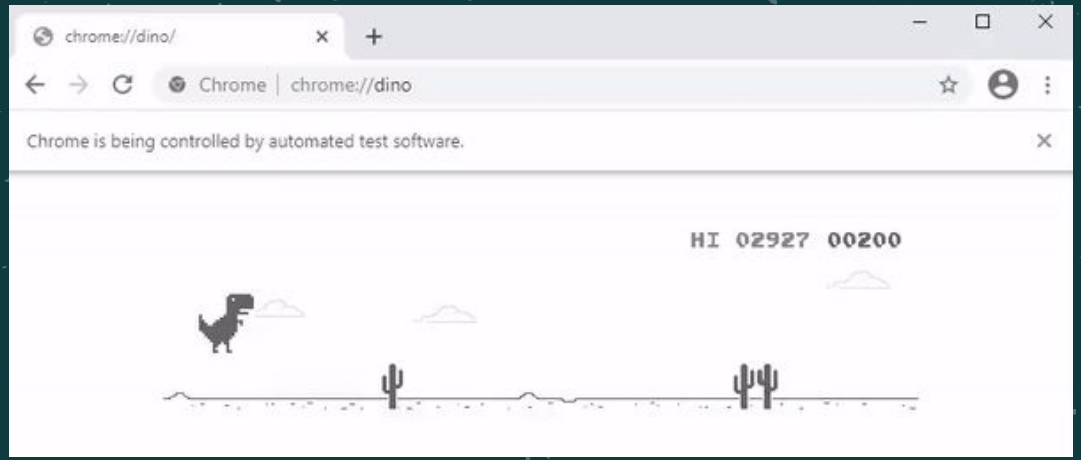
Initial Training Process

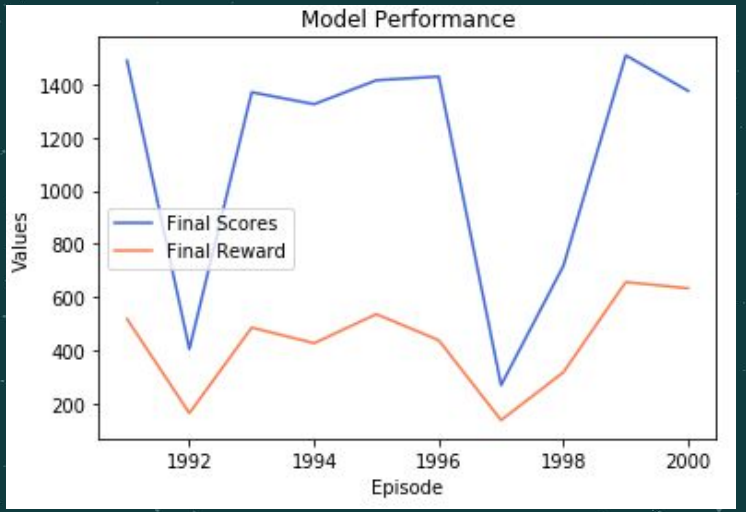




- In the first 10 episodes, the model performs badly and only managed to obtain a maximum score of 59. The initial reward is only ranged from 12 to 22.

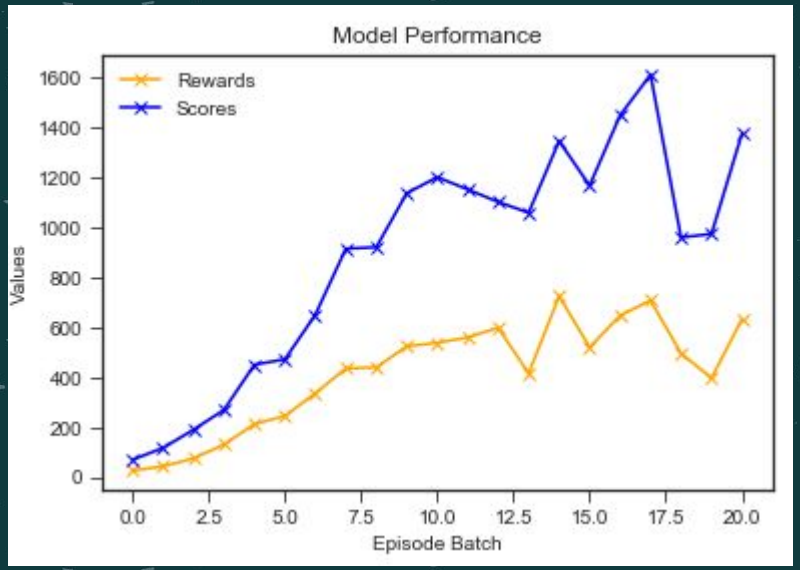
Model Performance





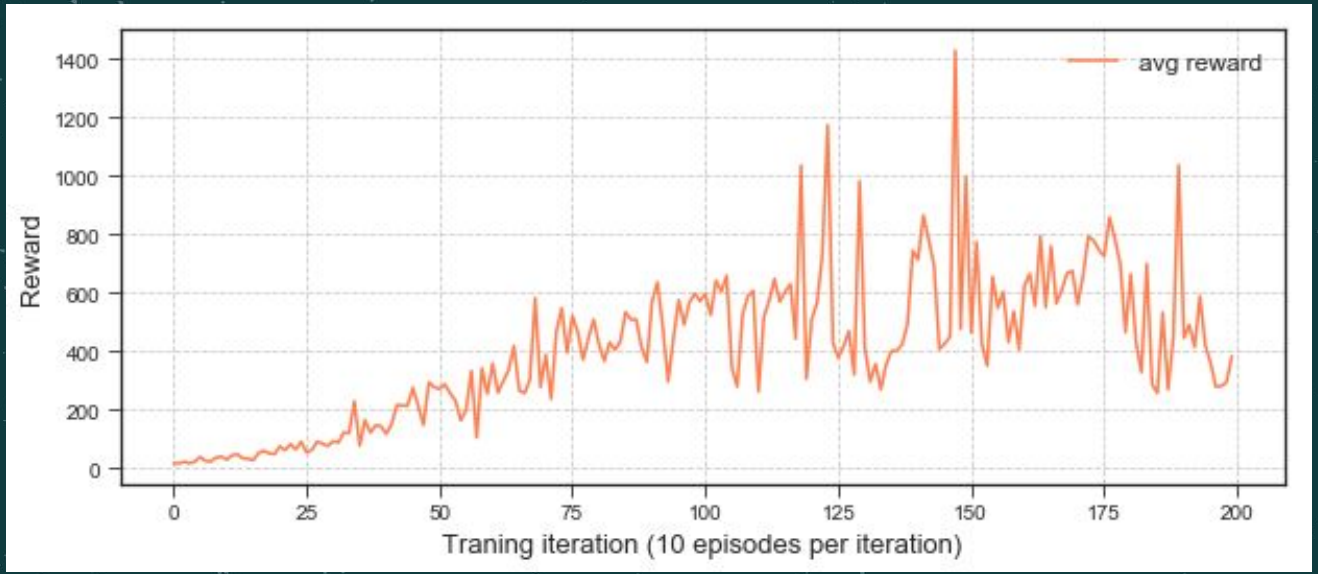
- During the training process, the model learns and the performance improves to be better. In the final 10 episode, the model obtained a maximum score of 1509 and the reward has reached 655.7

Model Performance for Each Episode Batch



- Each episode batch consists of 100 episodes.
- The model performance is improving linearly over episode bath in the first 10 episode batch. The same trend is observed for the total rewards.
- In the last 10 episode batch, the model performance is fluctuating and reach a peak scores in the 17th episode batch.

Average Reward



- The average reward value increased linearly over the training iteration.

Highest Score

- The agent obtained a highest score of 3028 on the 1489th episode
- The total reward is 952.1 with an epsilon value of 0.0001

Contributions

Member	Contributions
LIM JIA QI	Initiated and proposed ideas, model development and training
WONG HAO SHAN	Initiated and proposed ideas, model performance evaluation
CHONG SIN MEI	Model performance evaluation and result visualization
CHEAH JO YEN	Result visualization and prepare presentation slides

Thank You