Chrome-Dino Run Game

using Deep Reinforcement Learning

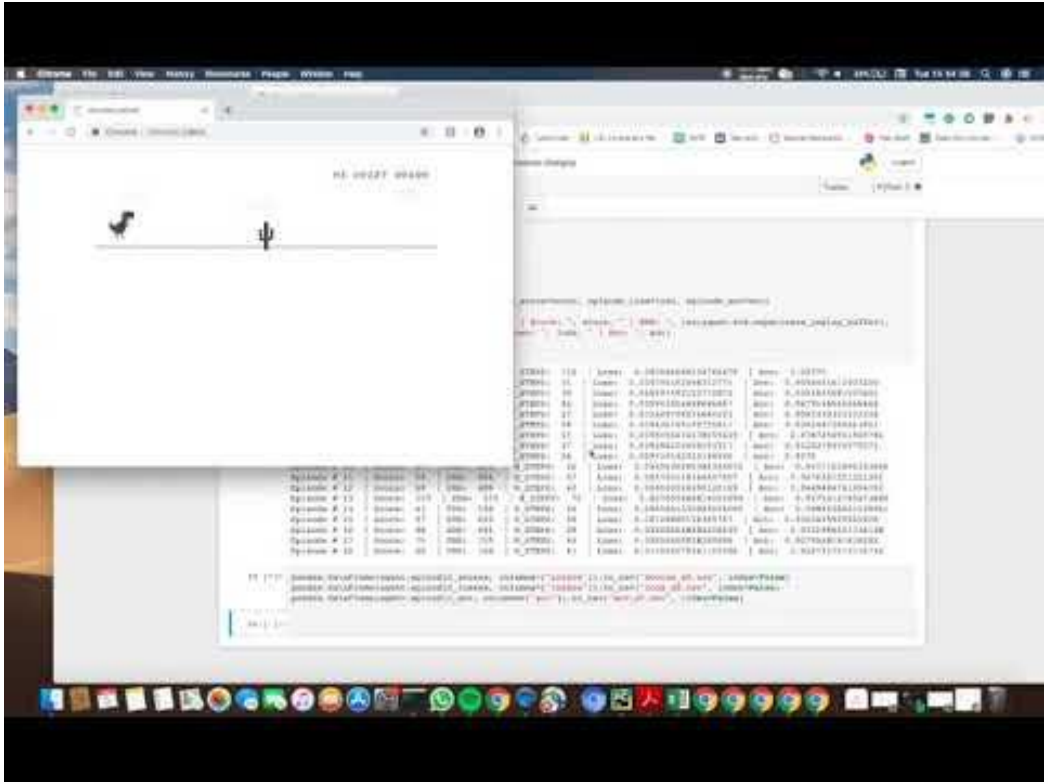
By Group #6

- Yogesh Jadhav
- Sameer Shinde
- Pooja Thakoor

Why we chose this project?

- Dino Run - an endless real-time runner game based on JavaScript in Chrome browser. It’s addictive and fun!
- Something fancier than conventional Supervised Learning!
- Exploring Reinforcement Learning

Let's have a look at the gameplay!

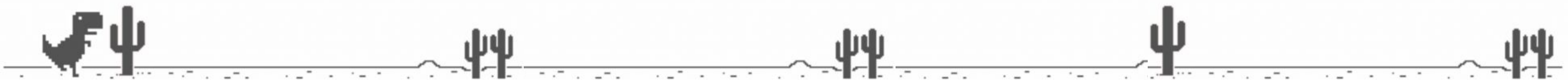


Inspiration and Tricks

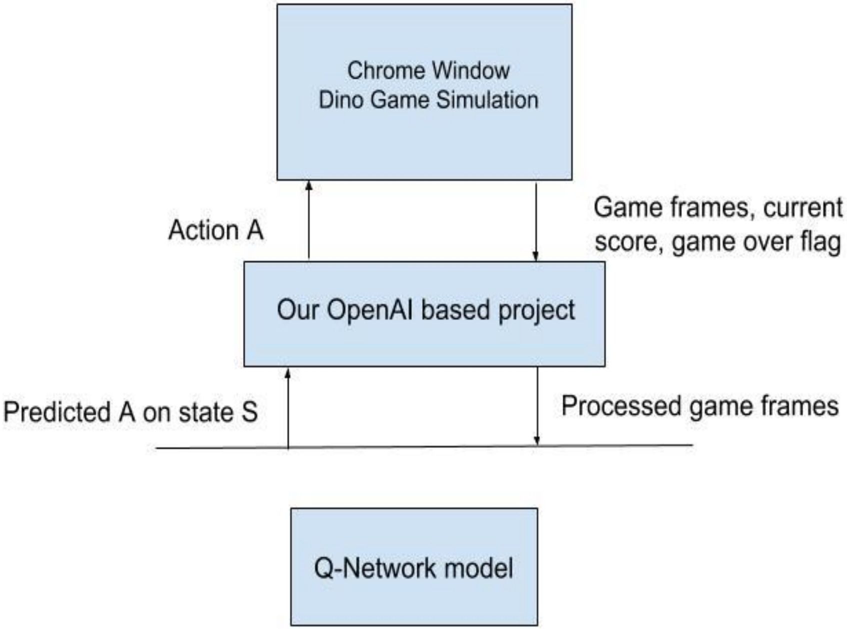
A 2013 publication by Google's DeepMind titled 'Playing Atari with Deep Reinforcement Learning'.

Key ideas from the paper:

1. Experience Replay Buffer (ERB): A data structure with fixed max size to store previous experiences which supports random sampling and deletion operations.
2. Random Exploration: A technique by which agent takes random actions initially to explore new states.



Project Structure



State

Current Snapshot of Game
[Image]

Actions

Jump (1)

Stay (0)

Reward

0.1 for every step

-1 for terminal state

Algorithm Overview

1. Have a empty ERB
2. Initialize Q-function approximator network with random weights
3. Repeat for all the games:
    a. Start a new game and observe the initial state S
    b. Repeat till the game is not over
        i. Select action A with RANDOM-EXPLORATION and carry out A on S to get S’
        ii. Observe reward R for taking action A on S
        iii. Store experience <S, A, R, S’> in ERB
        iv. S = S’
        v. Sample random experiences <s, a, r, s’> from the ERB.
        vi. Train the network on the MSE (New Q(s, a), Q(s, a)) loss.

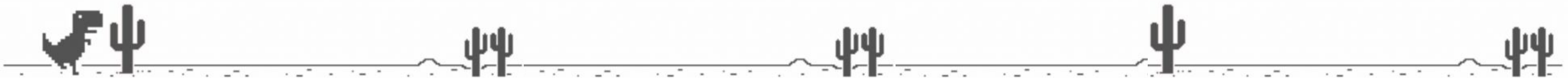
Algorithm Overview

$$\boxed{\text{New } Q(s,a)} = \boxed{Q(s,a)} + \boxed{\alpha}\left[\boxed{R(s,a)} + \boxed{\gamma}\,\boxed{\max Q'(s',a')} - \boxed{Q(s,a)}\right]$$

- New Q Value for that state and the action
- Learning Rate
- Reward for taking that action at that state
- Current Q Values
- Maximum expected future reward given the new state (s') and all possible actions at that new state.
- Discount Rate

Challenges we faced (and solved!)

- Frame drops - Small enough model to predict action A on state S faster. Smaller batch size to train.
- Learn the game speed: Stack 4 consecutive states together to create a state S.
- Converge faster - Guided start for the model.



Overview

- Image Capturing and Preprocessing
- Model Architecture
- Results

Image Capturing & Preprocessing

Image Capturing

- Two way interface between browser and model
- Open AI Gym Environment **gym-chrome-dino**

- Real time frame capture
- 80 x 160 image
- Unstacked & Stacked image

Unstacked Image

Stacked Image

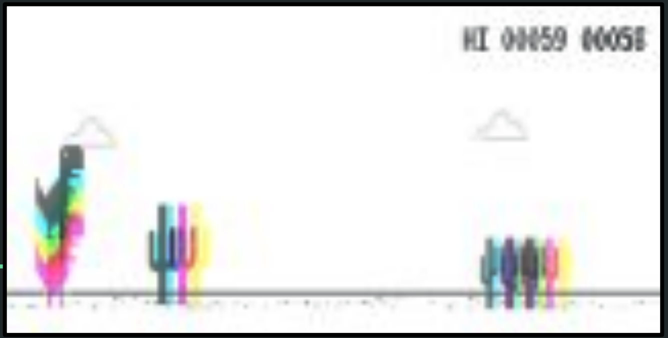


Image Processing

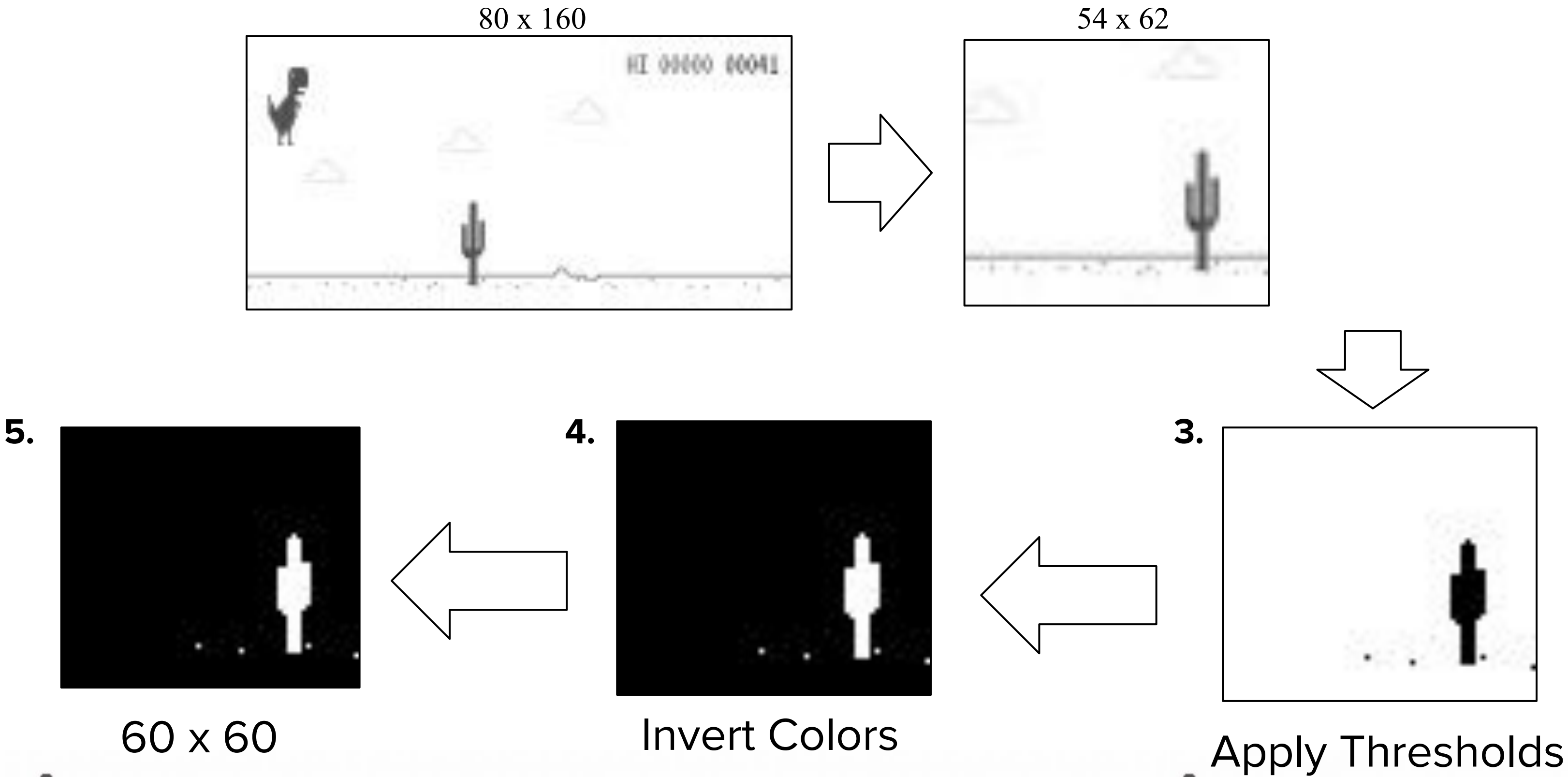
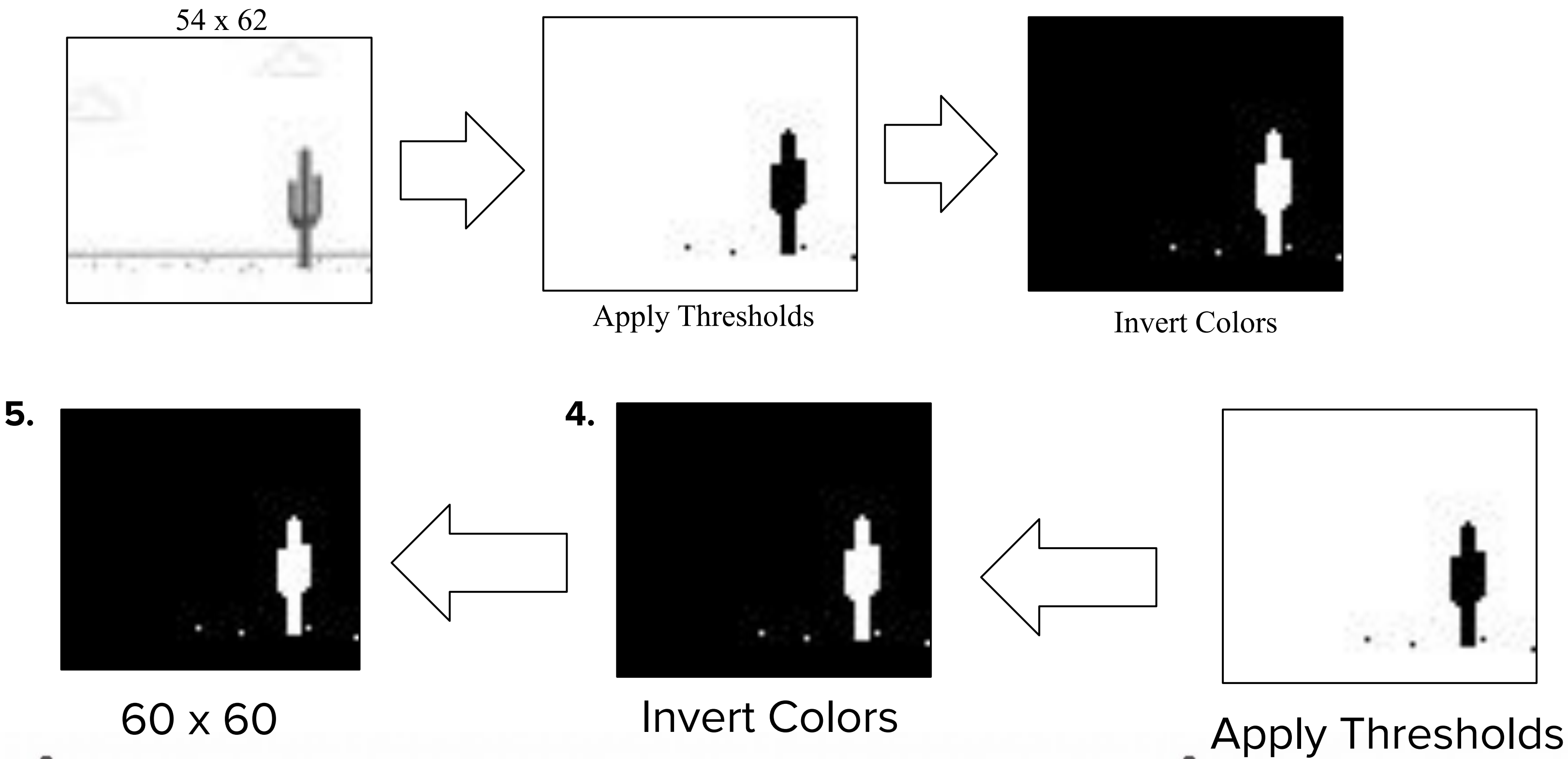
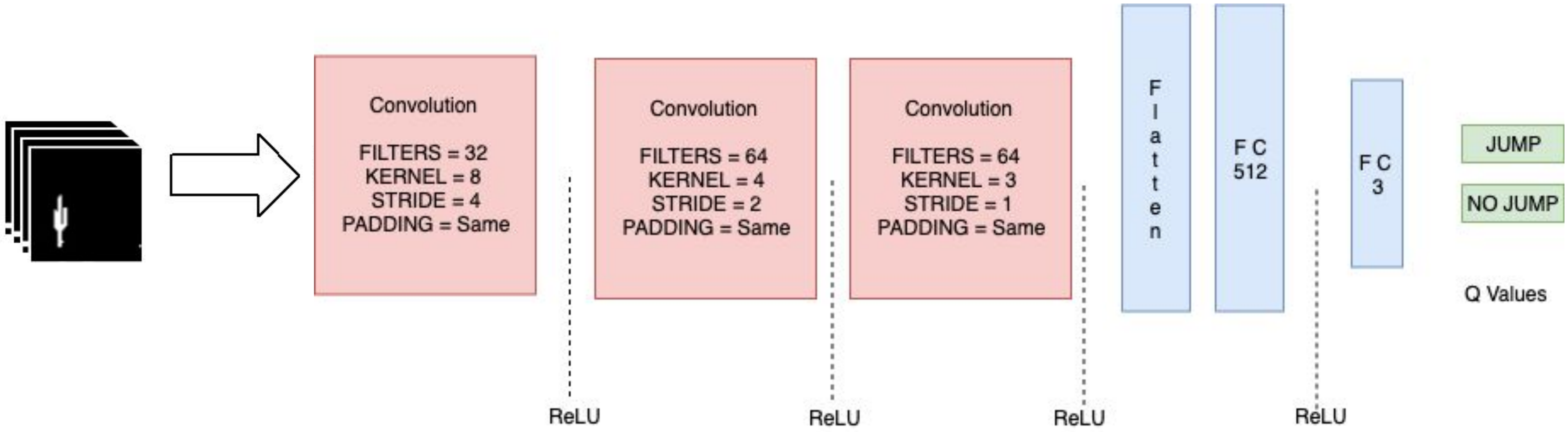


Image Processing

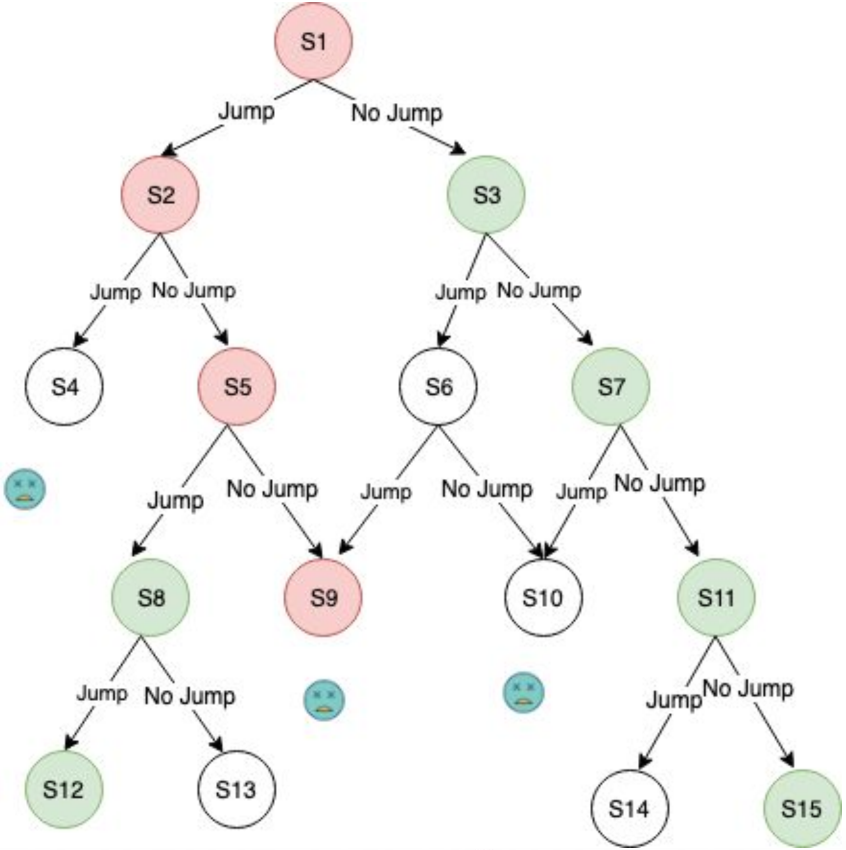


Model Architecture

Action prediction using Neural Network



Exploration-Exploit Dilemma



Random Exploration

Why:

- With certain probability, try to explore all states by Random Exploration
- Initially this probability has to be big
- As you progress, reduce this gradually

How:

TRAINING_FACTOR = CURRENT_ERB_SIZE / MAX_ERB_SIZE

adjusted_random_exploration =
RANDOM_EXPLORARION_INITIAL + (TRAINING_FACTOR *
(1 - RANDOM_EXPLORARION_INITIAL))
min(adjusted_random_exploration, RANDOM_EXPLORARION_MAX)

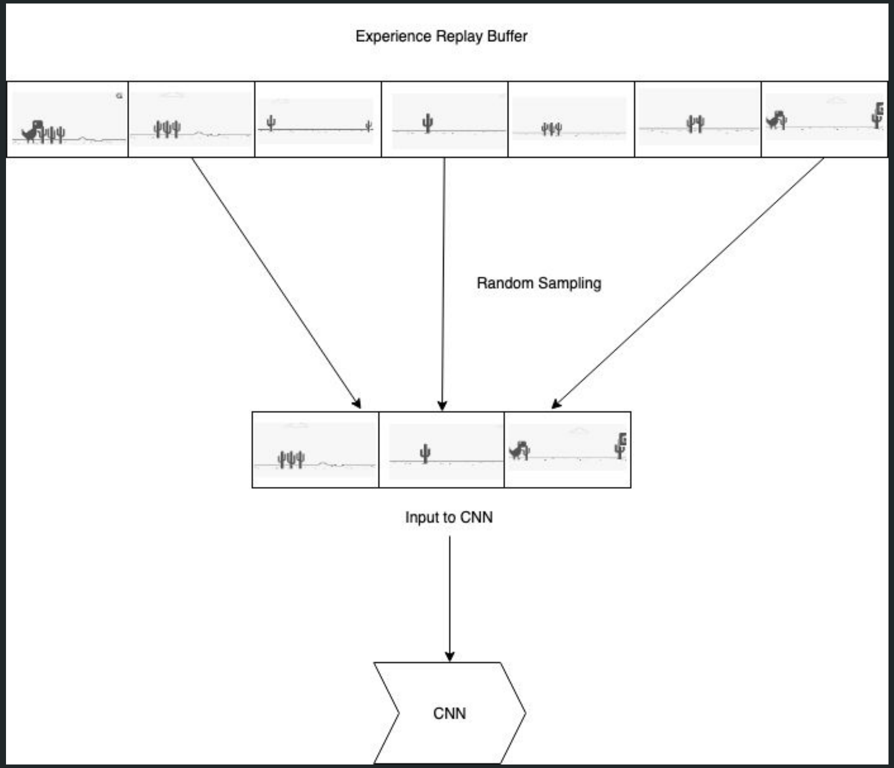
Experience Replay Buffer

Why

- Learn from previous gameplay
- Separate the simulation part from the training part.
- Provides “revision” of the past experiences to make model less forgetful to past learning.

How

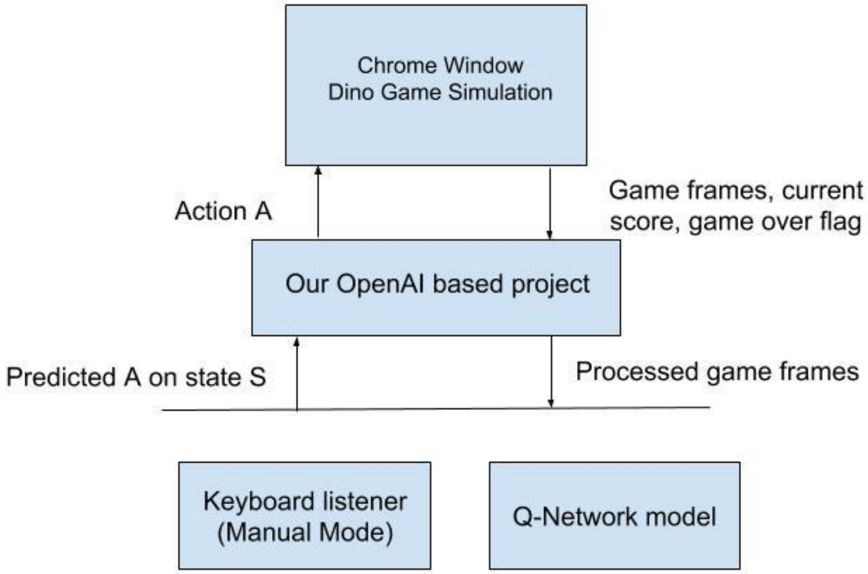
- Train randomly sampled experiences from ERB



Guided Start

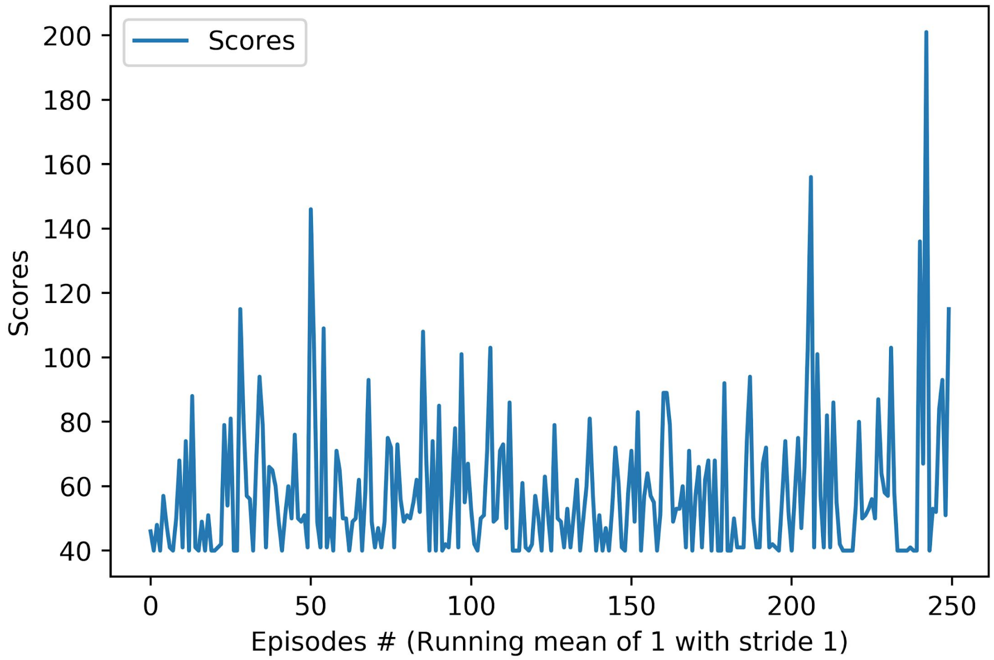
Seeding ERB with few correct experiences to start the training with!

Guided Start

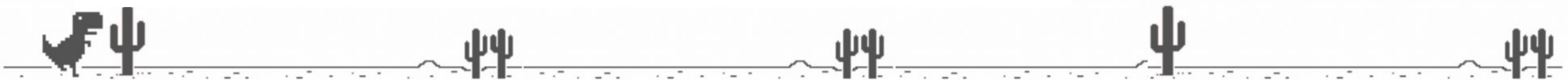


Results

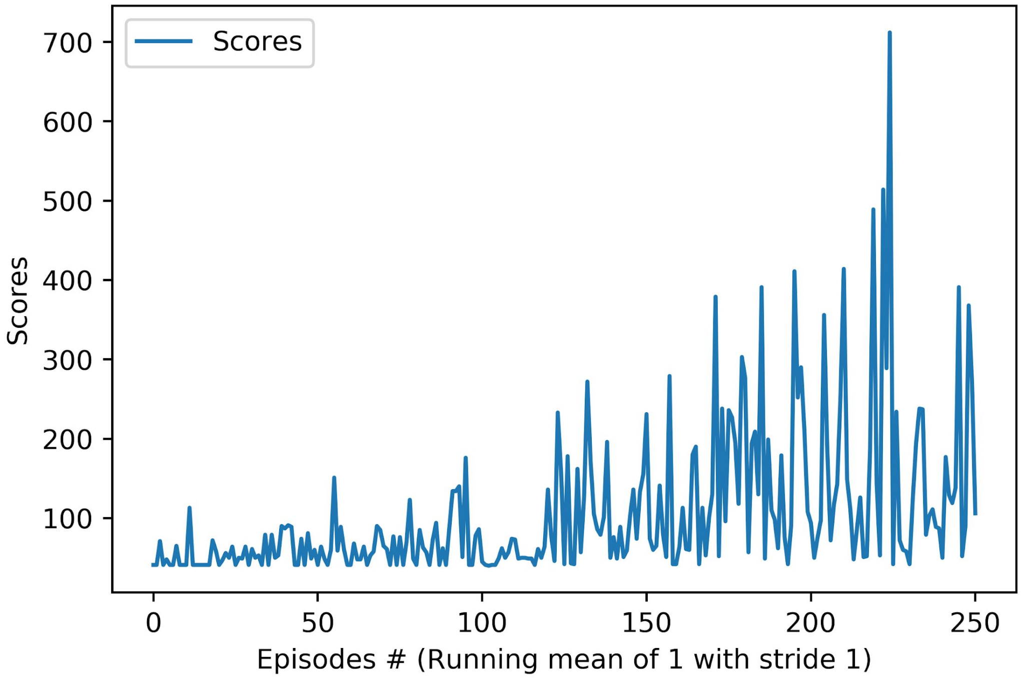
Scores over 250 episodes (Stacked Input)



- Max Score: 200
- Score increases gradually

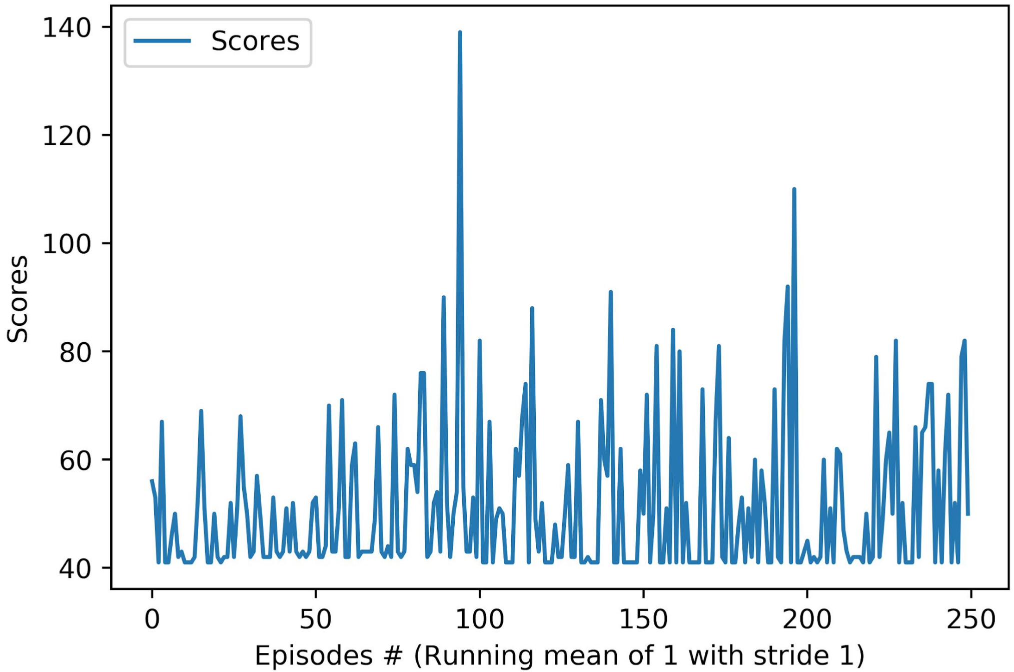


Scores over 250 episodes (Stacked Input with Guided Start)



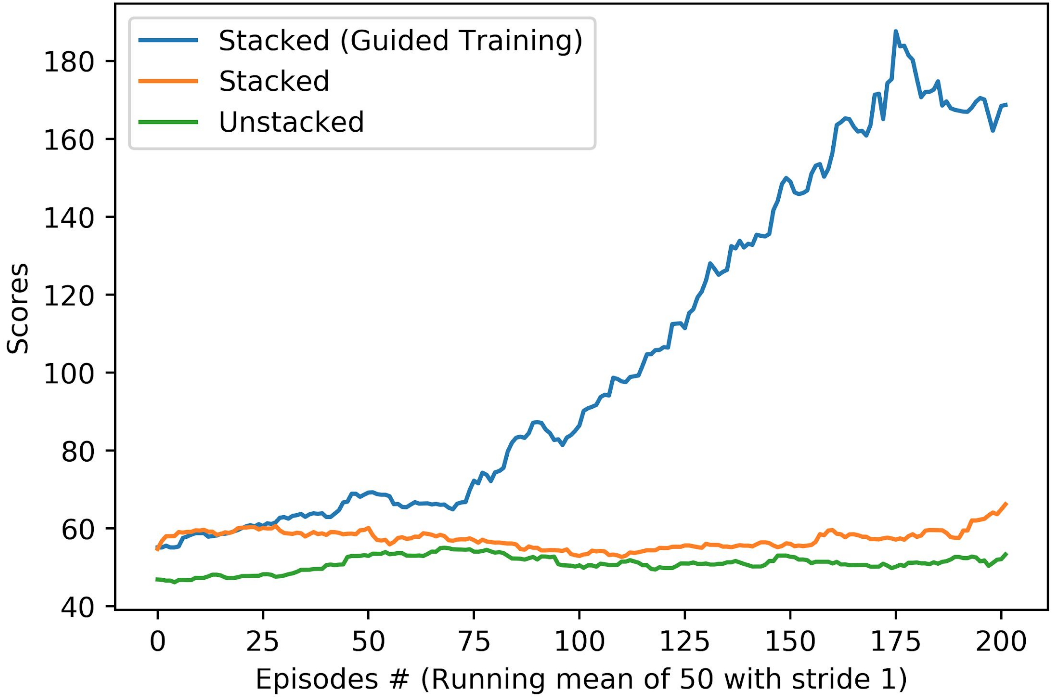
- Max Score: 712
- Learns Faster

Scores over 250 episodes (Unstacked Input)



- Max Score: 140
- Slow learning compared to stacked input

Observation

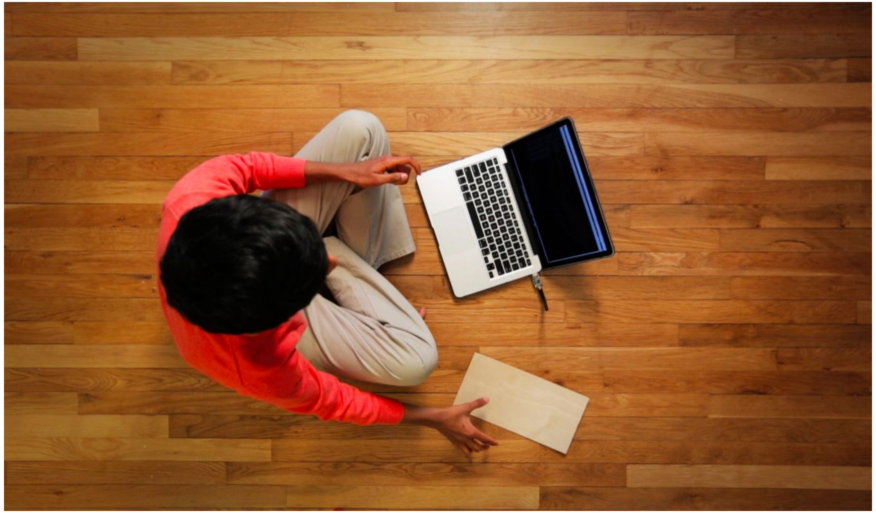


Stacked version learns better as compared to unstacked version as we train more

Experiments

What all things did we try?

1. Preprocessing stage
   - Different input image structure and sizes
   - Stacked / unstacked input
2. Architecture stage
   - Different hyperparameters and layers complexity
3. Guided start



Future Scope

Future Scope

- Long sighted model - use wider view of the gameplay for state by cropping less
- Additional Training - Training on more episodes

References

1. https://github.com/elvisyjlin/gym-chrome-dino/tree/master/gym_chrome_dino
2. https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
3. https://www.intel.ai/demystifying-deep-reinforcement-learning/

Thank You