Stanford University

Super Mario KAIrt: AI agent

Charles Tsao and Claire Mai, {chtsao, cmai21}@stanford.edu

Stanford | ENGINEERING
Computer Science

Introduction

- **Project Goal: Develop an AI agent to play Super Mario Kart using Reinforcement learning**
- OpenAI Gym Retro provides an Integration UI that we used to define variables and done condition. [1]

Integration UI


```
{
 "info": {
  "collision": {
   "address": 8261714,
   "type": "<u2"
  },
  "rank1": {
   "address": 8261696,
   "type": "|u1"
  },
  "speed": {
   "address": 8261866,
   "type": "<u2"
  },
  "surface_type": {
   "address": 8261806,
   "type": "|u1"
  },
  "timer": {
   "address": 8257794,
   "type": "<d4"
  },
  "wrong_direction": {
   "address": 8257800,
   "type": "<d4"
  },
 }
}
```

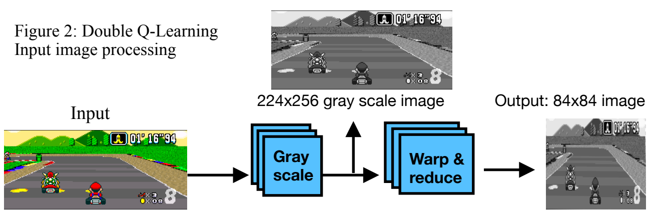

JSON file for variables

- Modeled the game as a state-based graph with states, actions, rewards, and next states.
- Implemented Q-learning with epsilon greedy and Double Q-Learning (a variation of DQN with 2 convolutional neural networks: one to determine the policy and the other to determine its optimal value) [2]

Data Acquisition

- Input: A frame of the game represented as a 224x256x3 tensor, corresponding to a 224x256 image with 3 RGB channels
- **Q-learning with epsilon greedy data acquisition:**
  - Converted 224x256x3 tensor to a vector of size 3 by averaging pixel values x axis wise (i.e. 224x256x3 frame → 256x3 tensor) and then averaging pixel values y axis wise (256x3 tensor → vector of size 3)
- **Double Q-learning/Q-learning (v2) data acquisition:**
  - Converted to gray scale then reduced size to 84x84 thereby warping the image. Process seen below in Figure 2:

Figure 2: Double Q-Learning Input image processing



References and Acknowledgements

- We would like to thank our CS221 mentor Horace Chu for all the support and guidance.

[1] OpenAI. "OpenAI Gym." *Gym*, gym.openai.com/.

[2] Van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. arXiv preprint arXiv:1509.06461, 2015.

Method/Implementation

Method 1: Q-learning with epsilon greedy (v2)

- Dictionary represented q-table. Each key mapped a state (encoded frame) to vector of 3 q-values, one for each action.
- Trained for 100,000 episodes (1 episode = 1 game of Super Mario Kart)

Learning rate: Alpha	Discount factor: gamma	Initial epsilon	Epsilon decay	Epsilon minimum
0.1	0.6	1	0.9999997	0.1

Convolutional Neural Network for Double Q-Learning:

- Deep Q Network shown in Figure 3
- Input: Four 84x84 stacked grayscale images (84x84x4)
- 3 convolutional layers using ReLU with (32 filters, 8x8, stride = 4), (64 filters, 4x4, stride = 2), and (64 filters, 3x3, stride= 1) respectively
- 2 fully connected layers (flatten and dense)
- Output: n q-values, each one corresponding to a certain action

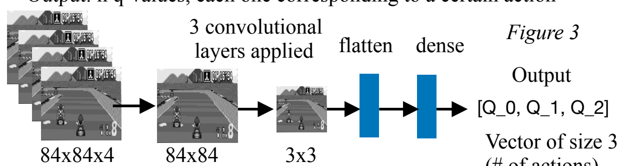


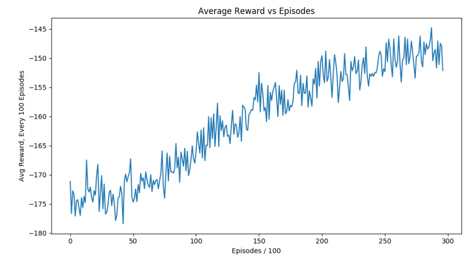
Figure 3

- **Method 2: Double Q-Learning**
- Implemented convolutional neural network using TensorFlow

Attempt	# of actions	ϵ	Epsilon decay	Done condition	# of episodes	Reward function
1	5	1	0.99999975	On dirt, exceed time limit, or finish race	10,000	-2*rank + (-100 if wrong direction else 0)
2	3	1	0.99999975	On dirt, exceed time limit, or finish race	10,000	-2*rank + (-100 if wrong direction else 0)
3	3	0.5	0.999999	Collided with obstacle, exceed time limit, or finish race	50,000	-2*rank + (-100 if wrong direction else 0) + (-40 if surface type is dirt, else 0
4 (v1)	3	1	0.9999993	Collided with obstacle, exceed time limit, or finish race	50,000	-2*rank + (-100 if wrong direction else 0) + (-40 if surface type is dirt, else 0
5 (v2)	3	1	0.9999995	Going in wrong direction, exceed time limit, or wins race	100,000	-2*rank + (-100 if wrong direction else 0) + (-20 if surface type is dirt, else 0) + (-10 if collides else 0)

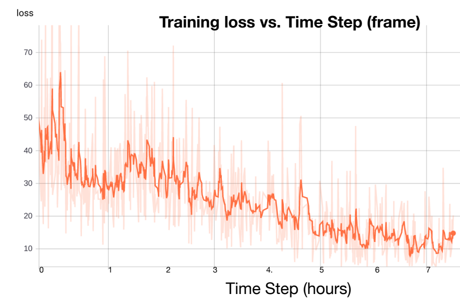
Results & Discussion

Q-Learning with Epsilon Greedy (v2) Results:

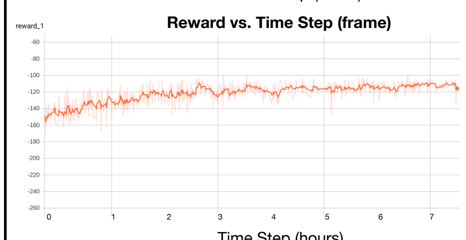


- Despite rise in avg reward per episode, agent did not learn
- Avg reward caps at -145
- Agent ends game early to achieve high reward

Double Q-Learning Results (v1):



- Q-Learning algorithm only looks one state ahead which leads to poor estimations of q-values
- CNN learns weights that predict optimal q-values



- Double Q Learning can retrieve more information for each state, thus able to predict q-values better
- Agent is able to complete 2-3 laps of the course

Future work

- Train for longer (i.e. 200 million frames), as discussed in a paper [2]
- Continue to fine tune hyper parameters
- Engage advanced techniques for image augmentation
- Explore NEAT algorithm to train AI agent based on generations
- Collect more data on other Super Mario Kart courses so the AI agent is able to play drive in all types of races