Beating OpenAI Gym with Deep Q Learning

Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

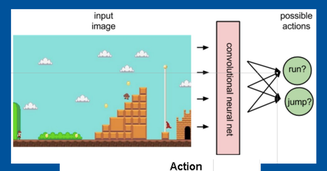
James Bocinsky and Lorenzo Tabares | Dr. Li | Big Data Ecosystems

Problem / Question

Design a general deep neural network trained with reinforcement learning that is able to beat multiple OpenAI Gym games.

Deep Q-Learning

- Q Learning
  - Table consists of:
    - Row vectors as states
    - Column vectors as actions
  - Updated using Bellman equation:
    $$Q(s,a) = r + \gamma(\max(Q(s',a'))$$
  - Maximizes immediate reward and future discounted rewards.
- Convolutional Neural Network implementation
  - Determines the optimal policy through approximation.
  - Minimizes the loss function to approximate the optimal value function

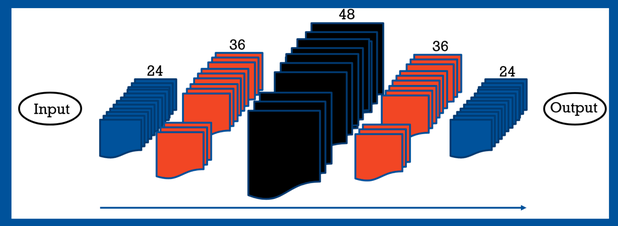


Action

$$R = \begin{array}{c|cccccc} \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

Project Overview

- Comparison of Neural Network Architecture Performances
- Reinforcement Learning and Deep Q-Learning
  - OpenAI Gym Environments
- TensorFlow + Keras
- Implemented Network Architecture
  - 3-layer Neural Network trained with Deep Q learning - > Neural Network A
  - 4-layer Neural Network trained with Deep Q learning - > Neural Network B
  - 7-layer Deep Neural Network trained with Deep Q learning - > Neural Network C



Open AI Gym

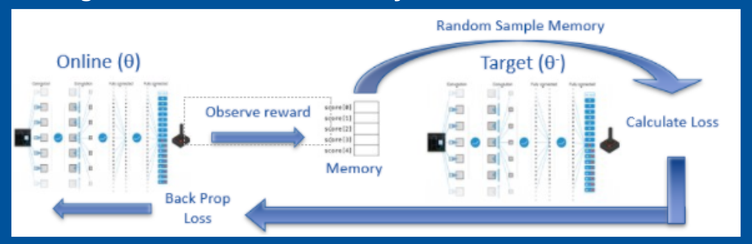
- Why did we choose OpenAI Gym?
  - Easy assessment of our progress
  - Compare with leaderboard performances
  - Add to community
- Solve a wide range of games with one agent
  - First step towards a generalized intelligence

Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

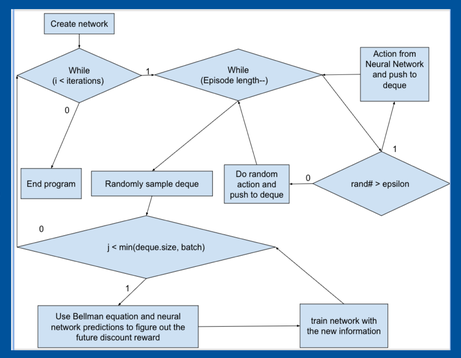
Dual Network Overview

- Two networks of the same architecture are used
- "Online" network – Used to form memory of previous actions and rewards given
- "Target" network – Uses memory to calculate loss



Procedure

- Two training Phases
  - Phase 1: Populating Dequeue
  - Creating memory for storing results of running network
  - Save: state, action, reward, next state in dequeue
  - Phase 2: Update Training with bellman equation
  - Mini-batch learning from dequeue memory using bellman equation to train



Game Enivornments

- Pole Cart(v0)
  - Input(observation space)
  - Output(Action Space)
  - Reward- if end game condition not met
    - Add 1 to score
  - End game conditions:
    - Pole tilt past 12 degrees
    - Cart moves off screen
    - Game has played 200 steps

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Num	Action
0	Push cart to the left
1	Push cart to the right

- Mountain Car(v1)
  - Input(observation space)
  - Output(Action Space)
  - Reward- if end game condition not met
    - Add 1 to score
  - End game conditions:
    - Car reaches 0.5 position
    - 200 iterations are reached

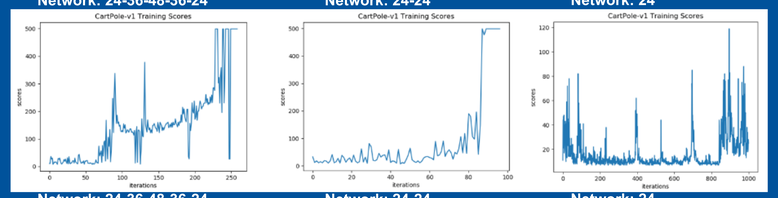
Num	Observation	Min	Max
0	position	-1.2	0.6
1	velocity	-0.07	0.07

Num	Observation
0	push left
1	no push
2	push right

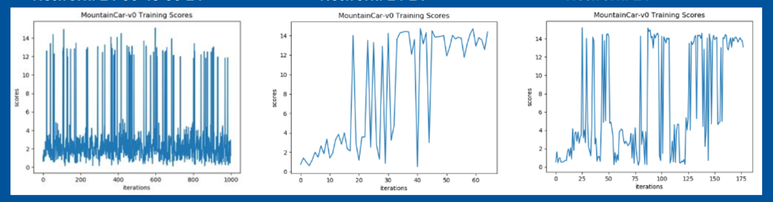
- Arcobot(v1)
  - Input(observation space)
    - Link speed
    - Link velocity
  - Output(Action Space)
    - Swing left
    - Swing right
    - No Swing
  - Reward
    - How high the second link is
  - End game conditions:
    - Car reaches 0.5 position
    - 200 iterations are reached

Results

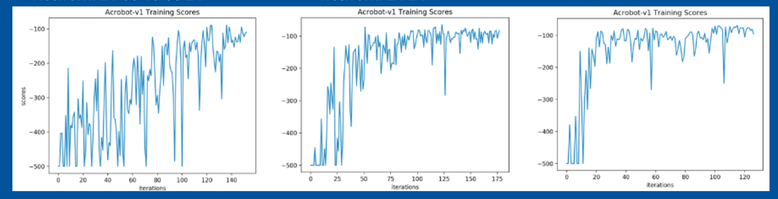
Network: 24-36-48-36-24 | Network: 24-24 | Network: 24



Network: 24-36-48-36-24 | Network: 24-24 | Network: 24



Network: 24-36-48-36-24 | Network: 24-24 | Network: 24



- Average final score over 10 games

Game:	Network A (24)	Network B (24-24)	Network C (12-24-48-24-12)
Pole Cart	38.1	500	500
Mountain Car	13.79	13.29	1.78
Acrobot	-94	-161	-385

Conclusion

- In conclusion, we can see that choosing a different architecture based on the problem at hand is very important when designing neural networks. For complex problems, a more complex network is desired but for simple problems, it may not be necessary. We can see through our analysis that of the three networks, network B performed the best. It was able to successfully beat all games 10 times in a row with its final weights. Network A was able to learn a policy for the simpler game of mountain car, which had a smaller environment space, and the acrobat game, but was not able to learn an optimal policy for the more complex game of pole cart. Similarly, network C was able to learn an optimal policy for the pole cart game, as this was more complex, but it was not able to learn an optimal policy for the simpler environment space of mountain car. Also, network C struggle with the arcobot game since it was undertrained.

Future Works

- Implement Convolutional Neural Network for Atari Games
- Implement Actor-Critic Model for pendulum and Continuous-Mountain-cart
- Implement Double Deep Q Learning for Atari Games