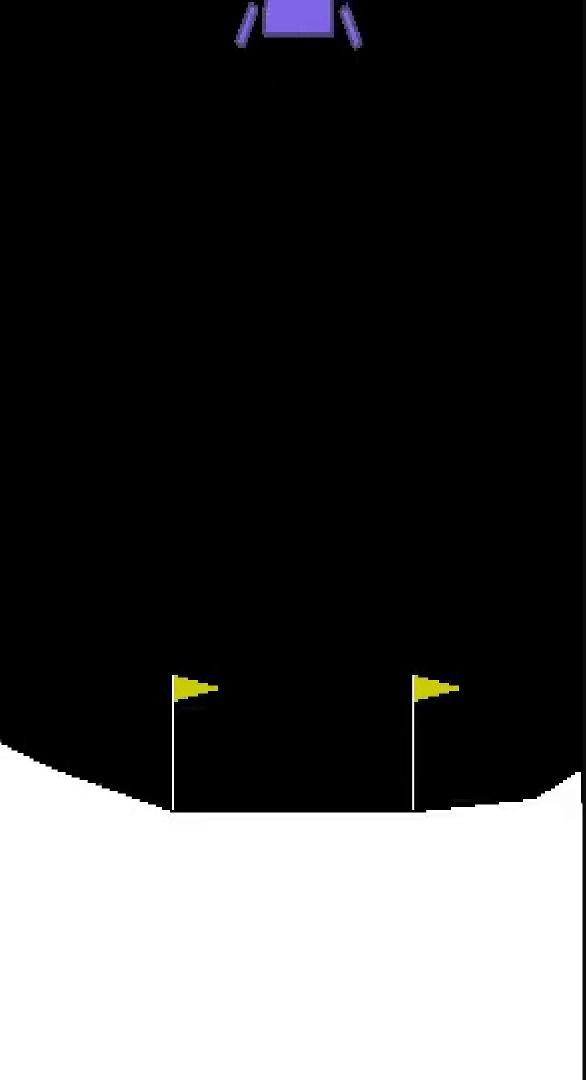
The Eagle Has Landed: RL with Lunar Lander

TIEN CHENG, FAQIH, ZHAO WU



Objective

ENVIRONMENT

Agent is a Lunar Lander that tries to land on a landing pad (0, 0). It knows it's coordinates, linear velocity in x and y direction, angle, angular velocity, and if it's legs are in contact with the ground.

ACTIONS

Agent can do nothing, fire it's main downward engine, or fire it's left and right RCS engines

OBJECTIVE

Train an agent that is capable of landing the Lunar Lander consistently and successfully.

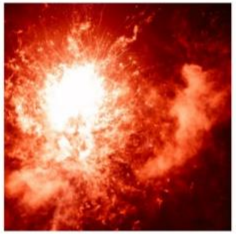
Reward Scheme

LANDING SUCCESSFULLY

+100 to +140 Points (Within Pad). All points lost if lander moves away from the landing pad.

FIRING MAIN ENGINE

-0.3 Points Per Step



CRASH

-100 Points

LANDING

+10 Points Per Leg

+100 For Soft Landing

Training

- **MODELS**

  Deep Q-Network, Double Deep Q-Network, SARSA

- **HYPERPARAMETER TUNING**

  Random Search was conducted for optimal learning rate, epsilon decay rate and target network update interval

- **TRAINING PROCESS**
  - Trained for 1000 episodes
  - Track average reward over last 100 episodes

- **EVALUATION PROCESS**

  Final models play 1000 episodes, and overall average reward, length and landings is recorded

- **BASE HYPERPARAMETERS**
  - Discount Factor, Gamma: 0.99
  - Batch Size: 64

Model Elaboration : Deep-Q Network (DQN)

- **USES NN TO ESTIMATE Q-FUNCTION**

  The optimal action is selected based on action with maximum q-value.

- **EPSILON GREEDY STRATEGY**

  Perform Exploration-Exploitation trade-off with decaying epsilon value.

- **UPDATE Q-VALUES BASED ON TD-ERROR**

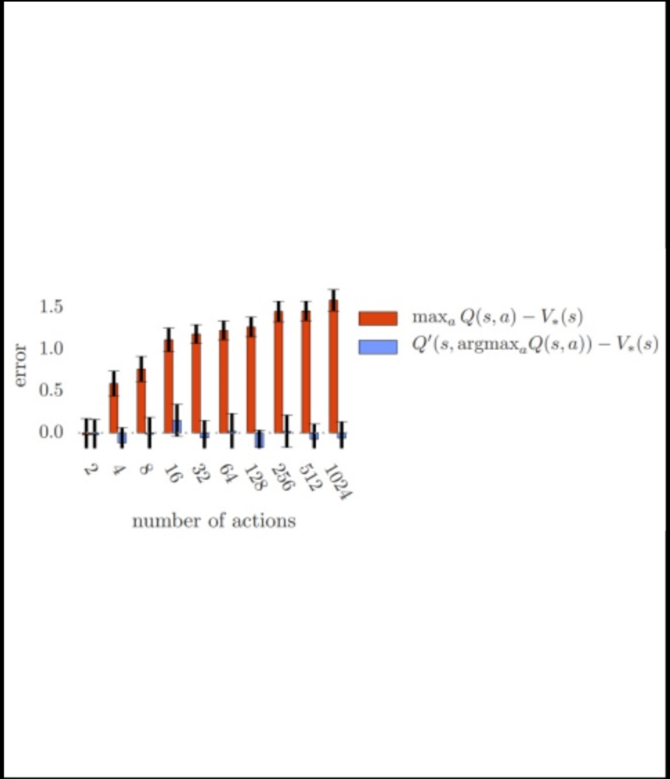
  Minimize the squared difference between the Expected Discounted Cumulative Reward (i.e. Bellman's Equation) and the one Generated from NN.

Model Elaboration : Double DQN (DDQN)

- **Q-VALUE OVERESTIMATION**

  DQN has tendency to overestimate Q-Values, resulting in suboptimal policy.

- **DECOUPLING Q-VALUE ESTIMATION AND ACTION SELECTION**
  - Online Net: Action Selection
  - Target Net: Q-Value Estimation for Bellman Update



Model Elaboration : State-Action-Reward-State-Action (SARSA)

ON-POLICY ALGORITHM

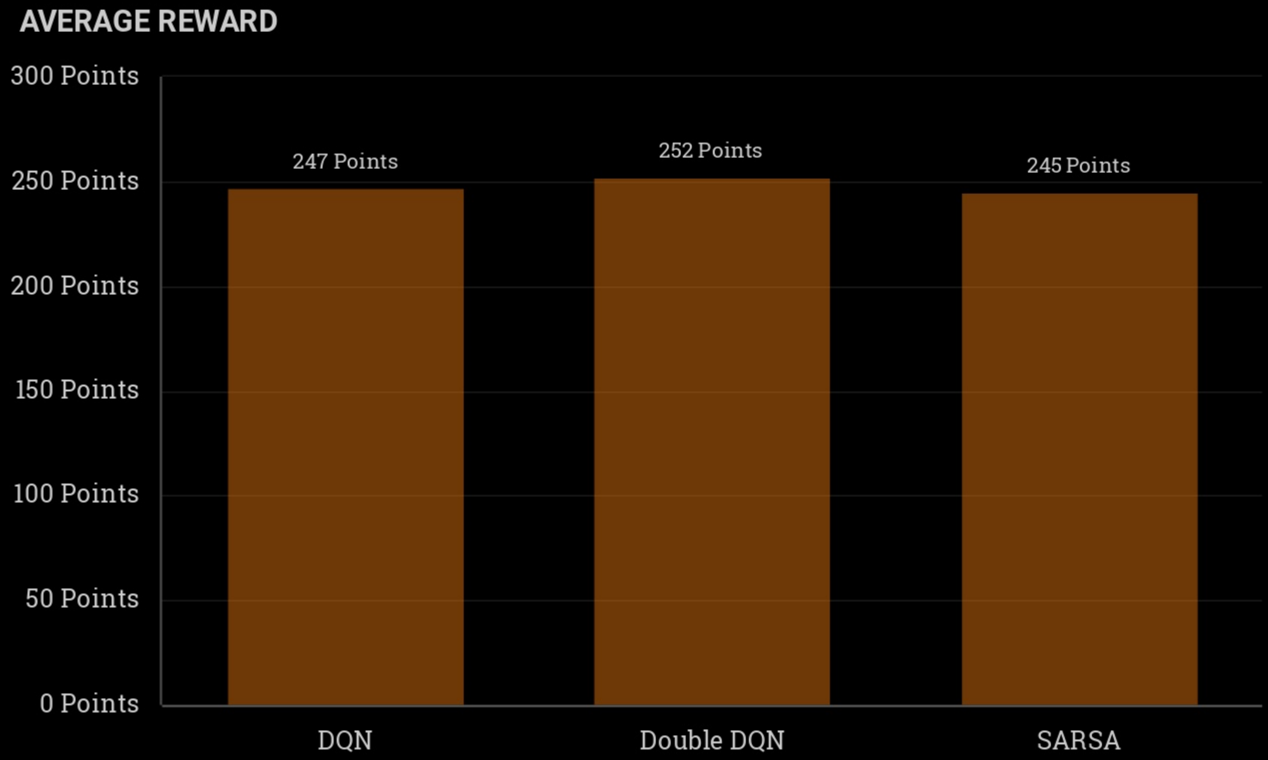
The same policy (with exploration and exploitation) function is used to select the next action during the TD-update phase.

PLAYING SAFE

- By penalizing the model even during exploration (i.e. choosing random action), the algorithm will tend to choose a safer action to achieve the goal than a more riskier ones

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma\, Q(s', a' < selected\ with\ \epsilon >)\ - Q(s,a)]$$

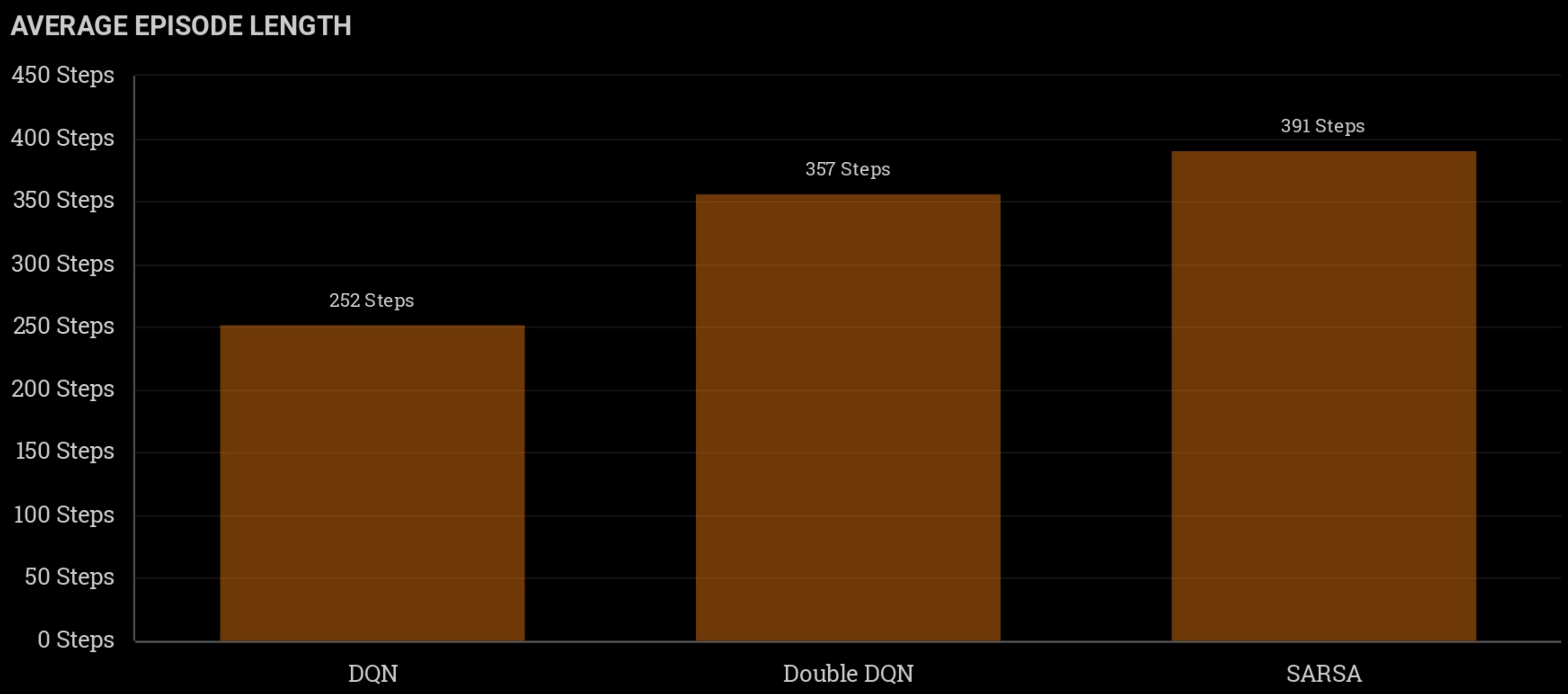
How Did The Models Perform?



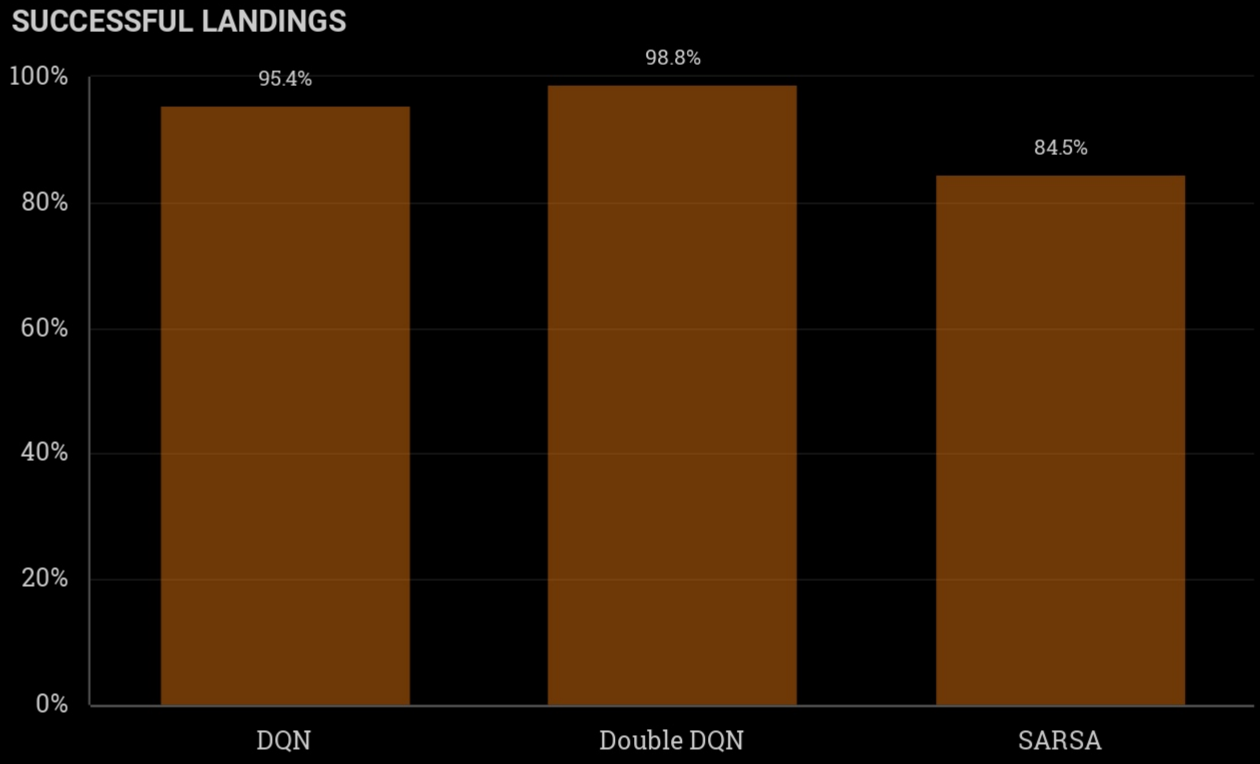
BEST MODEL:

DDQN

How Did The Models Perform?



How Did The Models Perform?



BEST MODEL:

DDQN


DQN 1/3

Conclusion

BEST MODEL

Double DQN performed better than the other models

DDQN PERFORMANCE

Appears to perform cautiously, taking longer to land

FUTURE IMPROVEMENTS

Introduce further improvements to DQN: PER, Dueling Architecture, Noisy Exploration