Playing Space Invaders and Q*bert using Deep Reinforcement Learning

Shreyash Pandey, Vivekkumar Patel

Stanford University

Objectives

To apply various techniques in Deep Reinforcement Learning to play two atari games: Space Invaders and Q*Bert.

1. Simple DQN with Experience Replay.
2. Implement Double DQN and Dueling DQN.
3. Implement DRQN.

Games

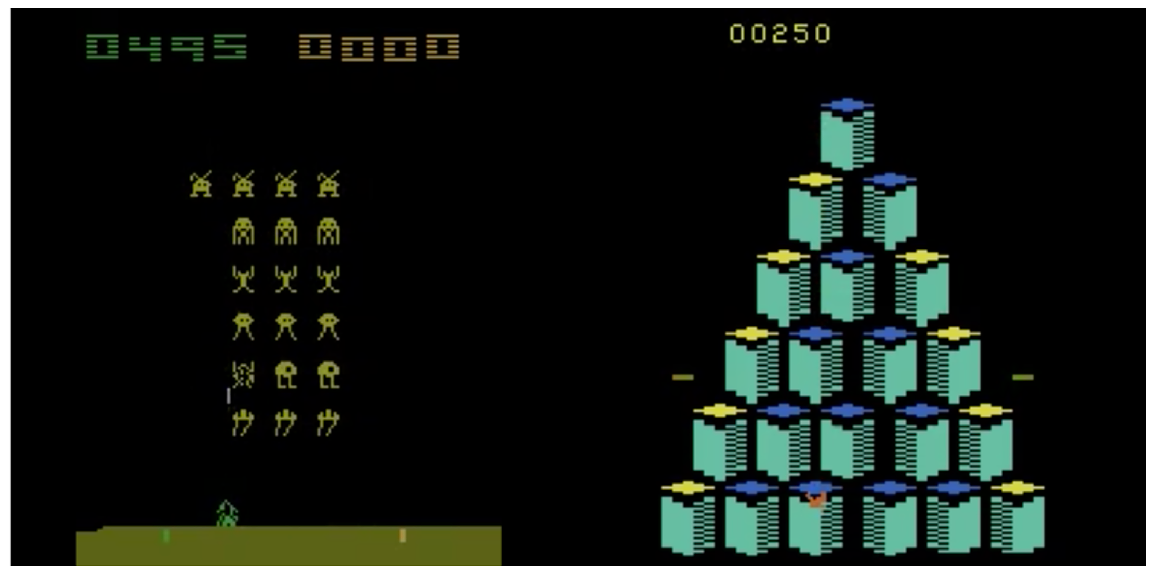
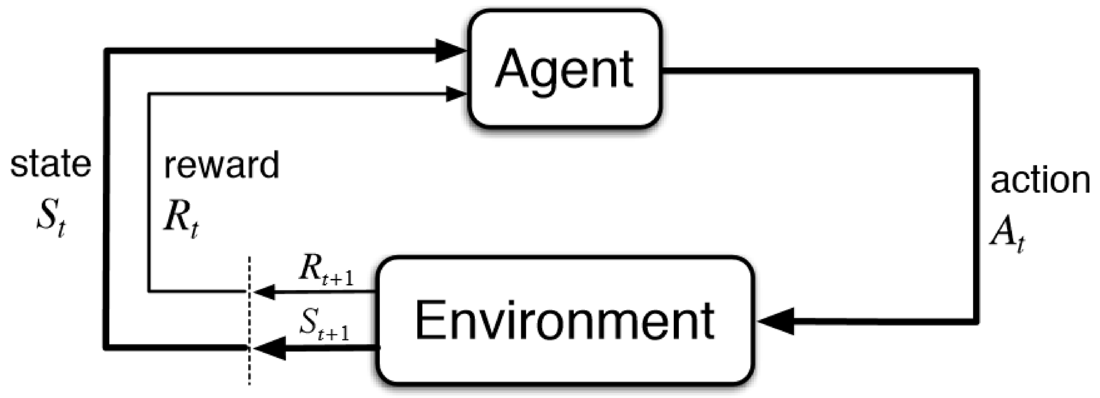


Figure 1: Space Invaders and Q*bert

Introduction

We are trying to build agents that learn to play Space Invaders and Q*bert. Our agents interact with the game enviroment through a sequence of observations, actions and rewards. We use the OpenAI gym Atari Emulator to emulate the environment for these games.



- State s: array of pixel values representing the image frame at that instance.
- Action a: All the actions possible for the agent to take.
- Reward r: Reward returned by the environment for that action.

DQN

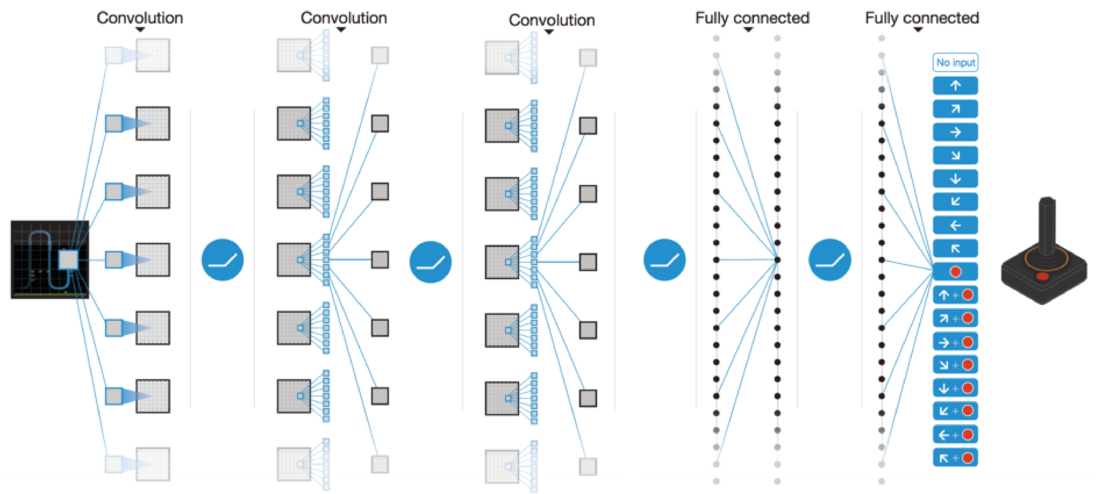
- Using a deep convolutional network provides flexibility to the model.
- Convolutional Layers focus on extracting good features from the image.
- Fully connected layers evaluate optimality of actions based on the features from below layers.
- $Y_t^Q = R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t^-); \theta_t^-)$

Experience Replay

- Store experience samples **(s,a,r,s')** in a buffer.
- Uniformly sample a batch of experiences to train.
- Advantages: Break Temporal Correlations, Reuse old data, use hardware-efficient minibatches.

Experimental Details

Network Architecture:

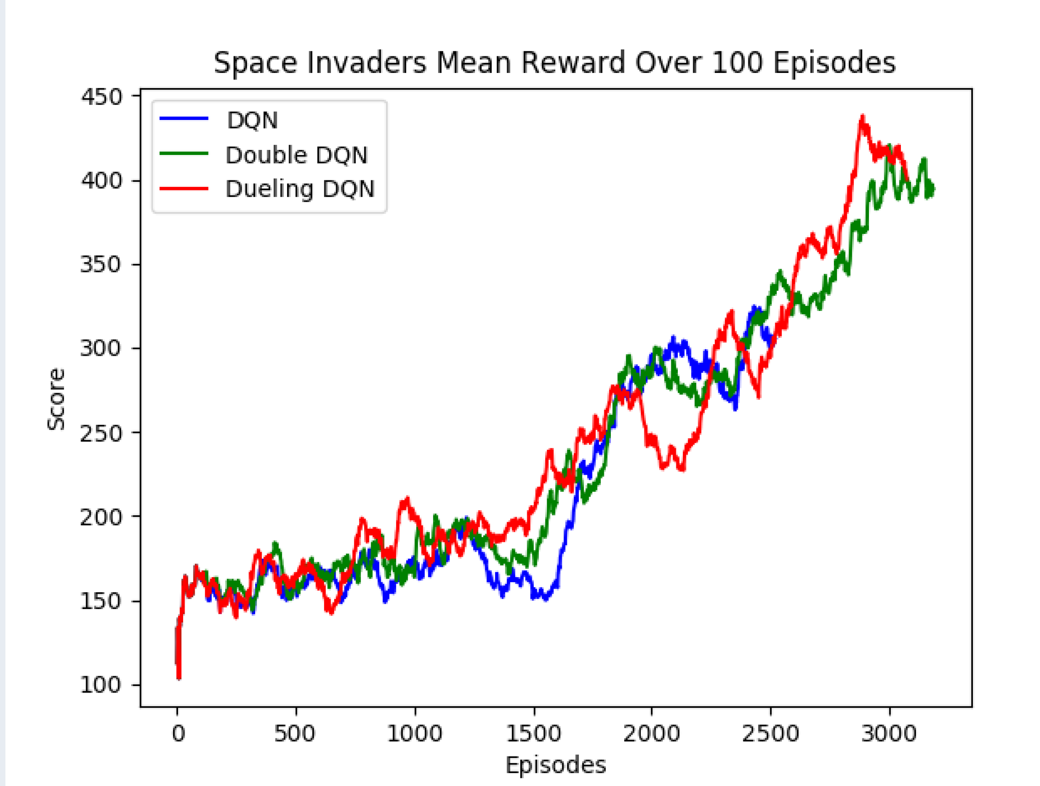


- Gray Scale Images of size 84x84.
- Replay Memory Size: 1M.
- We use Atari40M spec as emulator for these two games.
- We use a target Q network to generate the target Q values. We periodically update by equating this to the online Q network.
- For the Double DQN, we simply use the target Q network to evaluate actions.

Double DQN

- Overestimation of Q-values can lead to suboptimal policies.
- Solution: Decouple the selection and valuation of the actions.
- $Y_t^{DoubleQ} = R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta_t^-)$

Comparison on Space Invaders

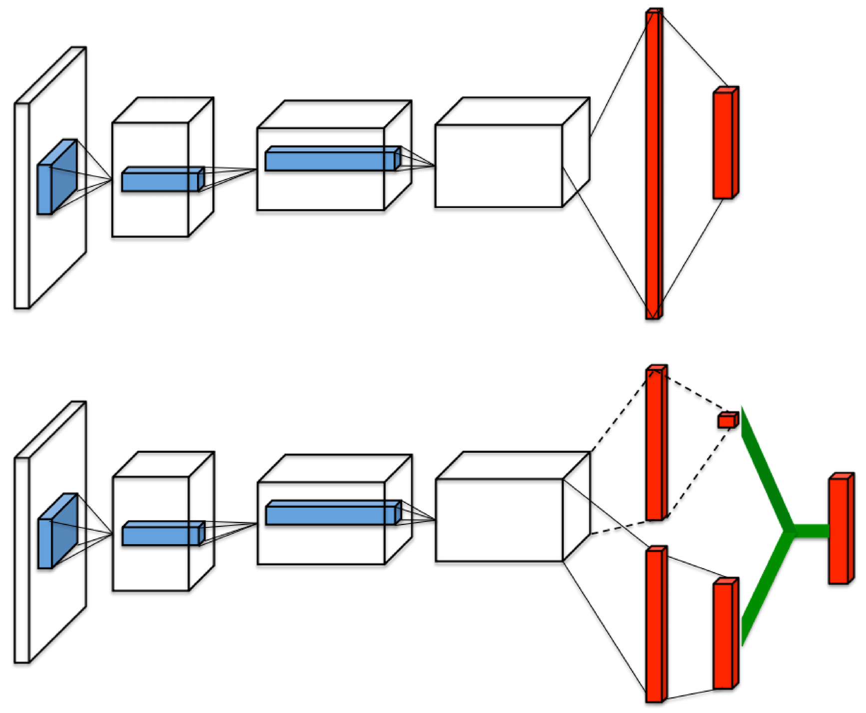


Analysis

- Performance: Duelling DQN > Double DQN > Vanilla DQN.
- Dueling DQN achieves more robust estimates of state value by decoupling it from action values.
- Experimented with different loss functions - Huber loss, and clipped Bellman error - clipped Bellman error lead to faster convergence and better results.
- Experimented with using DeepMind RL wrappers such as episodic life that help value estimation.

Dueling DQN

- Model state value function and **advantage function** separately.
- Advantage function defined as $A(s, a) = Q(s, a) - V(s)$
- Value function is $V(s) = \max_a Q(s, a)$
- When bootstrapping in reinforcement learning, it helps to have a good estimate of V(s), independent of the action



Results

Algorithms	Max Scores
Vanilla DQN	324
Double DQN	422
Dueling DQN	440

Table 1: Scores on Space Invaders

Games	Baselines	Max Scores
Space Invaders	240	440
Q*bert	180	740

Table 2: Comparison Against Baseline

Future Work

- Compare performance of DQN, Double DQN and Dueling DQN on Q*bert.
- Implement DRQN and test the performance on both games.