CS419: Course Project

Mastering Atari using Deep Reinforcement Learning

Team Rage Quit

Anupam Nayak, Darin Jeff, Ishan Kapnadak, Sheel Shah

IIT Bombay

Table of contents

Motivation

Motivation

One of the long-standing challenges of reinforcement learning (RL) is to learn to control agents directly from high dimensional sensory inputs such as vision and speech.

Until very recently, most solutions relied on hand crafted feature representations to operate on these domains.

However, the performance of such RL systems depends heavily on the quality of feature representations used.

Motivation

A recent breakthrough in this field was made by *Volodymyr Mnih et. al.* of DeepMind Technologies in their paper titled “Playing Atari with Deep Reinforcement Learning”.

This paper presented the first deep learning model that successfully learned to control policies directly from high dimensional sensory input using reinforcement learning.

The goal of our project was to implement a few variants of the solution proposed in this paper, and to explore further advancements and improvements.

Experiment Design

Experiment Design

We explored two variants of the Deep Q-Network (DQN) model that was proposed in the paper.

1. *Double DQN*
2. *Dueling DQN*

These models were trained and tested on the popular Atari game, Pong. We used PyTorch for implementing the model and OpenAI Gym for implementing the game environment.

Experiment Design - Deep Q Networks

The central idea in Deep Q networks is to maximize the expected long term discounted reward (Q-value). This is done by finding $Q(s, a)$ - the expected long term discounted reward, given the present time step, when the state of the environment is s and the agent takes action a. This allows us to find the optimal policy, π^* which chooses the optimal action,

$$a^* = \underset{a \in A}{\arg\max}\ Q(s, a)$$

to maximise the Q-value.

Experiment Design - DQN v/s Double DQN

In simple Deep Q Networks, the function, $Q(s, a)$ is incrementally learnt by iteratively solving the equation

$$Q(s_t, a_t) = R_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a)$$

where γ is the discount factor. However, since the Q-values change each iteration, the convergence to the true Q-values is slow. Moreover, since we pick the maximum of $Q(s_{t+1}, a)$ among all actions, this leads to overestimation of the Q-value. This is overcome in *Double* Deep Q Networks by maintaining two copies of the same Q-network - Q_{local}, which is updated every iteration and Q_{target}, which is updated with the values from the local network every few episodes. Hence, for each iteration, we solve

$$Q_{\text{local}}(s_t, a_t) = R_{t+1} + \gamma \max_{a \in A} Q_{\text{target}}(s_{t+1}, a)$$

Experiment Design - Dueling DQN

A dueling DQN has a model such that the output Q function is decomposed into two components, as follows.

$$Q(s, a) = V(s) + A(s, a)$$

Here, $V(s)$ represents the *value* of the state s, which is how good it is to be in state s, and $A(s, a)$ represents the *advantage* of performing action a while in state s, which is how much better it is to take action a while in state s, as opposed to all other actions. A dueling DQN model can learn and estimate both $V(s)$ and $A(s, a)$ separately. By decoupling this estimation, our dueling DQN model can learn which states are valuable *without* learning the effect of each action at each state.

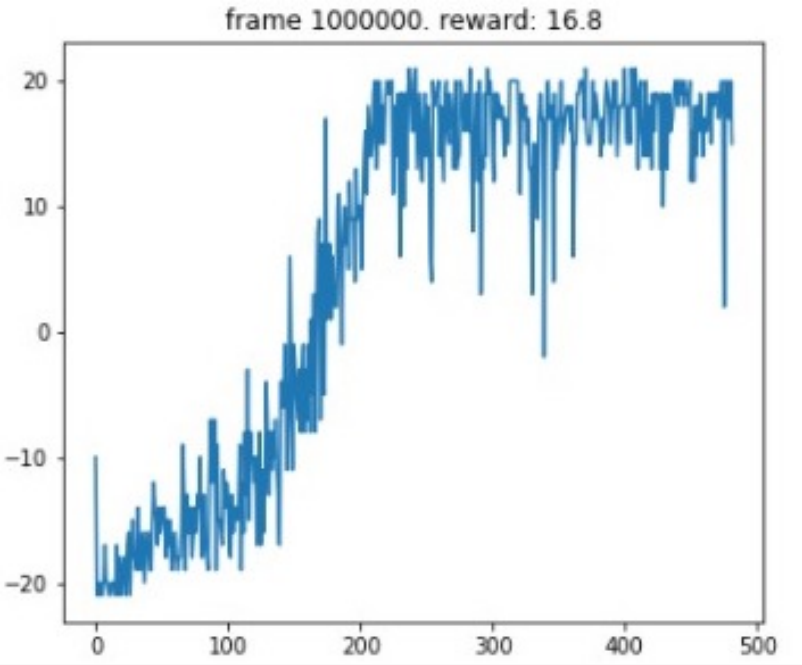
Experiment Design - Architecture

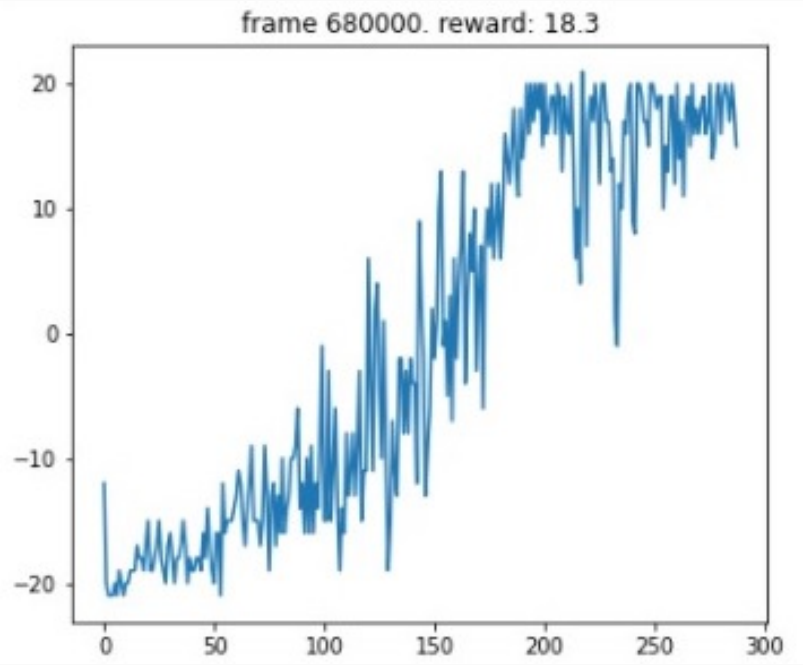
In both the variants, double DQN and dueling DQN, we implement a deep convolutional network that processes the game screen provided by the game environment. Next, we implement a full-connected neural network layer that uses the features processed by the convolutional network to learn the *Q*-function. The dueling DQN implements two parallel fully-connected networks - one network for the value function and one network for the advantage function.The detailed architectures for both the variants have been included in the report.

Results

Results

The results obtained for the double DQN (left) and the dueling DQN (right) are shown below.





The reward signal for the game of Pong is defined as the agent's score minus the opponent's score. We see that as the model is trained, the expected reward increases and saturates at 21, which is the maximum attainable reward.

Future Work

Future Work

There have been a lot of recent advancements in the field of reinforcement learning, with a lot of ideas derived from theoretical neuroscience. We wish to implement two of these advancements in our future work.

Future Work - Episodic Control

An obvious drawback of the Deep Q Learning model we have implemented is that the agent requires several orders of magnitudes more interactions with the environment as compared to a human to learn the optimal strategy. One method to overcome the above limitation is to design an agent that can implement *Neural Episodic Control*. Such an agent is able to quickly latch on to the optimal scenarios as soon as they are experienced, and hence learns the optimal strategy in a much smaller number of interactions with the environment.

Artificial agents employing episodic control show striking gains in performance as compared to deep RL networks, as is demonstrated by *Alexander Pritzel et. al.* in their paper titled “Neural Episodic Control”.

Future Work - Monte Carlo Tree Search

Another drawback of the Deep Q Learning model is that it operates mostly in a reactive or a trial-and-error way, where it learns by carrying out actions, observing the reward obtained and then deciding the optimal strategy. In contrast, humans learn in a simulation-based method. We use predictions generated from an internal mental model learned through experience to estimate the future rewards and then select the optimal action.

A lot of recent AI research has explored the avenue of simulation-based planning and model-based RL. One class of such methods is that of *Monte Carlo Tree Search Methods* which has led to a striking gain in performance at the game of Go, as demonstrated by *David Silver et. al.* in their recent paper titled “Mastering the game of Go with deep neural networks and tree search”.

References

References

- Volodymyr Mnih et. al. “Playing Atari with Deep Reinforcement Learning”, 2013.
- Deep Convolutional Neural Networks
- Arnav Paruthi. “Playing Atari using Reinforcement Learning”, 2019.
- Jacob Chapman and Mathias Lechner. “Deep Q-Learning for Atari Breakout”, 2020.
- Alexander Pritzel et. al. “Neural Episodic Control”, 2017.
- David Silver et. al. “Mastering the game of Go with deep neural networks and tree search”, 2016.

Acknowledgements

Acknowledgements

We would like to thank our instructor, Prof. Abir De and Ashish Tendulkar, of Google. This project would not have been possible without their guidance throughout the course. We would also like to thank the Teaching Assistants of this course for providing us with much-required assistance. We would like to thank our friends, Adit Akarsh and Sumeet Kumar, for their ingenious inputs and insightful suggestions. Lastly, we would like to thank our parents and family for their constant support and motivation through these trying times.

Thank you

Thank you!