Playing Atari Games With Reinforcement Deep Learning

Varsha Lalwani (11787)　　Masare Akshay Sunil (12403)

Prof. Amitabha Mukherjee

The Problem

Create an agent that can learn to play Atari 2600 games using Reinforcement Learning.

Motivation

For many years, it has been possible for a computer to play a single game by using some specially designed algorithm for that particular game. But these algorithms were useless outside their context. In this project we are trying to implement a deep reinforced learning based agent to play multiple video games. Also, if a General Game Playing system is well designed, it can be used in other areas as well, like search and rescue.

Past Work

Various previous methods to learn controlling agents have utilized a range of neural network architectures and have exploited both supervised and unsupervised learning. But using reinforcement learning for learning to control agents directly from high-dimensional sensory inputs like vision and speech wasn't much successful till 1995 when TD-gammon, a backgammon playing program which learnt entirely by reinforcement learning and self-play, achieved a superhuman level of play[1]. The first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning was presented by Mnih et. al.

Our Approach

We are using the reinforcement learning approach as it has been shown to yield better result than the supervised or unsupervised learning approach. In this approach can be divided into three major parts:

1. Convulational Neural Networks
2. Q-Learning
3. Emulation Interface

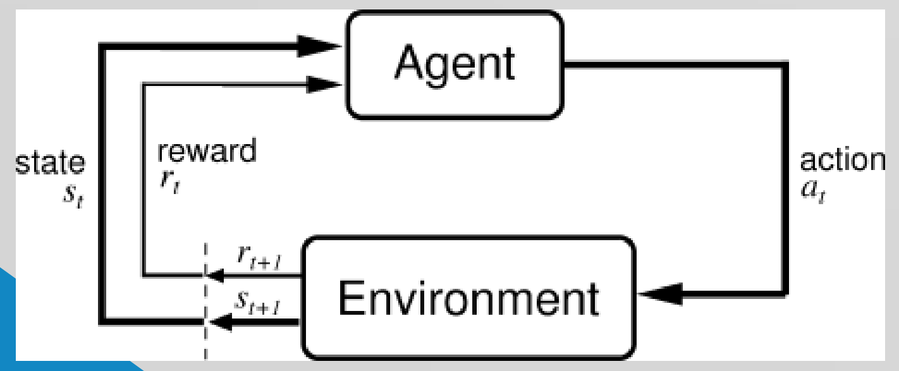
Our Algorithm

As, our approach in this project is quite similar to that of Mnih et. al. where we are using a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards

We can broadly describe our working algorithm as follows:

- *Initialize the game Emulation Environment Interface*
- *Take the screenshots of the game*
- *Pre-process the screenshots*
- *Use CNNs to extract the features from the screenshots*
- *Choose any action from the list of possible actions according to current state*
- *Observe reward and save it to memory*
- *Repeat and Train*

The basic model for reinforcement learning[1]



Convulational Neural Networks

Suited for extracting features from images

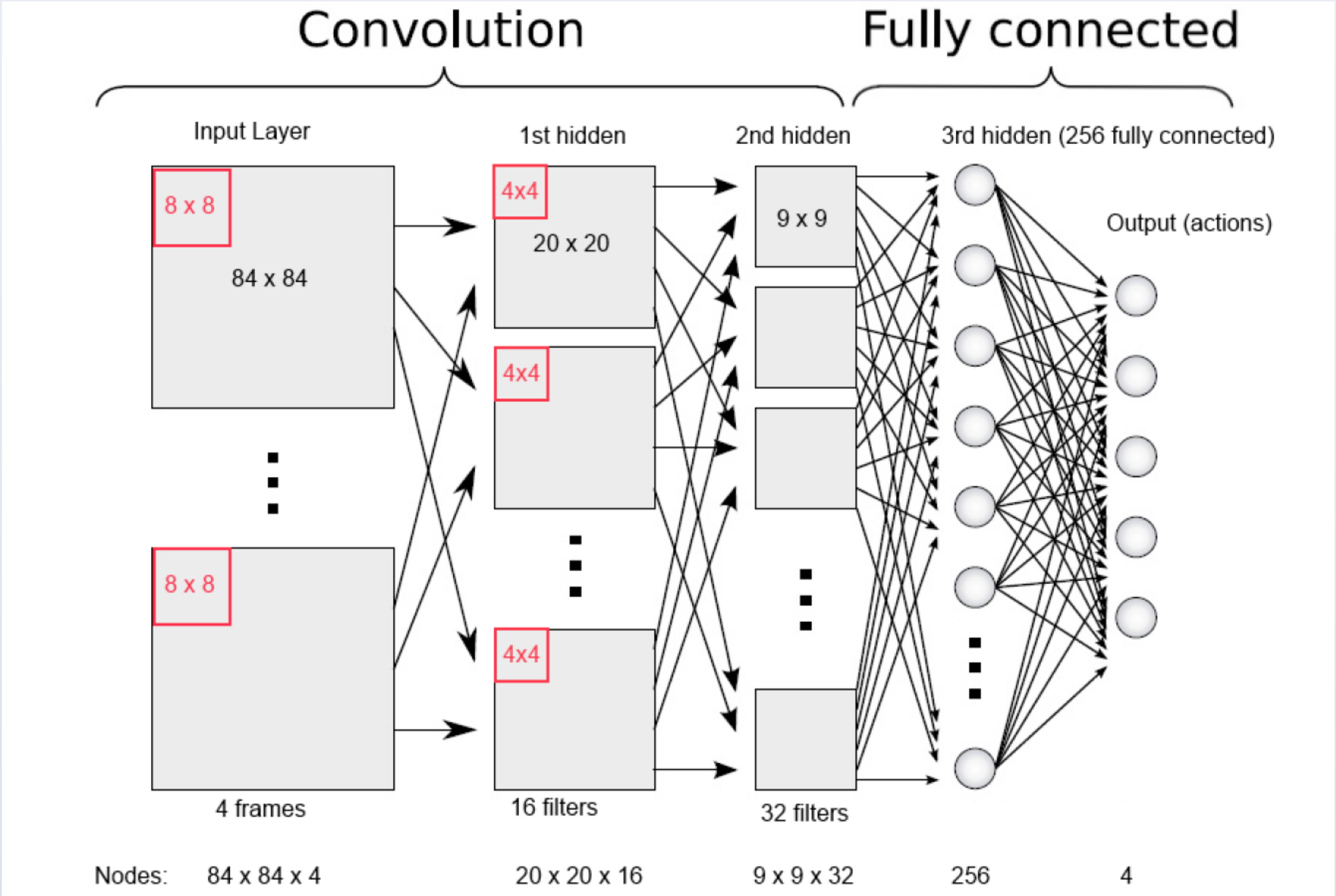
We take 4 images at a time

Images taken as 2D matrices

2D matrices convolved with linear filters

Weight matrices for multiple image

The CNN shown in the figure treats each image as a 2D object and convolved them with linear filters to get the state of the game. It also assigns the expected reward value to each possible action.



Q-Learning

Reinforcement learning: In a reinforcement learning model, an agent takes actions in an environment with the goal of maximising a cumulative reward. The basic reinforcement learning model consists of: a set of environment states S; a set of actions A; rules of transitioning between states; rules that determine the scalar immediate reward of a transition; and rules that describe what the agent observes.

Q-learning is a model-free form of RL algorithm:

```
controller Q-learning (S,A,γ,α)
 Inputs:
      S is a set of states
      A is a set of actions
      γ the discount
      α is the step size
 Local:
      real array Q[S,A]
      previous state s
      previous action a
 initialize Q[S,A] arbitrarily
 observe current state s
Repeat
    select and carry out an action a
    observe reward r and state s'
    Q[s,a] ⟵ (1-α) Q[s,a] + α(r+ γmaxa' Q[s',a'])
     s ⟵s'
until termination
```

The agent maintains a table Q[S,A], where S is the set of states and A is the set of actions.

At a given time, Q[s,a] is the current estimate of Q*[s,a].

Q*[s,a] is the expected value (cumlative discounted reward) of doing a in state s and then following the optimal policy.

Update: Q[s,a] ⟵ (1-α) Q[s,a] + α(r+ γ maxa' Q[s',a'])

It is an off-policy method i.e. learns the optimal policy no matter which policy it is carrying out.

Arcade Learning Environment

It is built on top of Stella, open-source Atari 2600 emulator

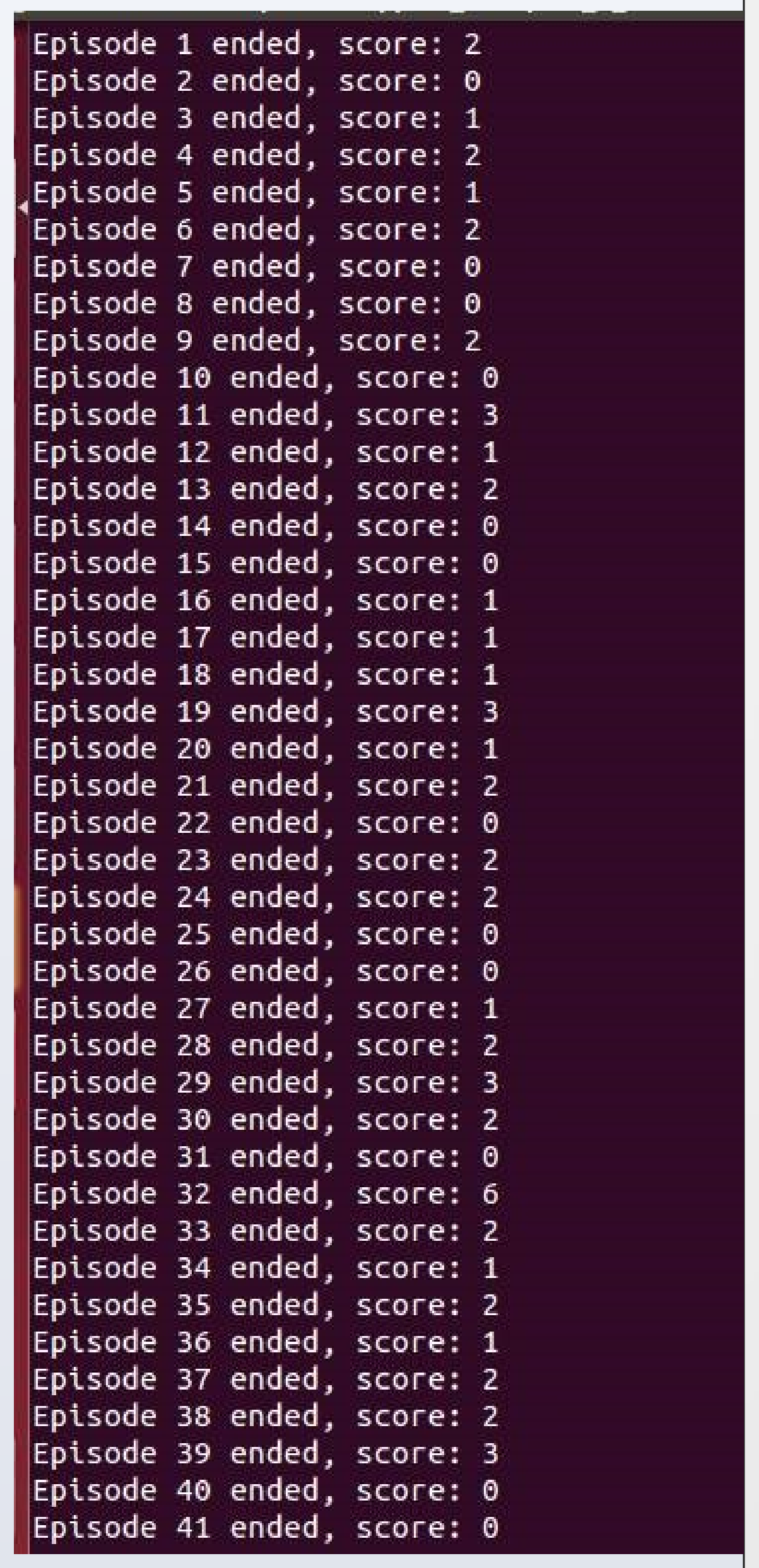
Built in C++, Support for over 50 games

Can programmatically input player commands

Outputs Image of the game screen, score and the state of the game

Why Breakout?

- Has only 2 states - dead or alive
- Has only 2 actions - left or right
- Need to reward only for alive states and very large negative reward for dead state.
- And it's FUN!!



```
Episode 1 ended, score: 2
Episode 2 ended, score: 0
Episode 3 ended, score: 1
Episode 4 ended, score: 2
Episode 5 ended, score: 1
Episode 6 ended, score: 2
Episode 7 ended, score: 0
Episode 8 ended, score: 0
Episode 9 ended, score: 2
Episode 10 ended, score: 0
Episode 11 ended, score: 3
Episode 12 ended, score: 1
Episode 13 ended, score: 2
Episode 14 ended, score: 0
Episode 15 ended, score: 0
Episode 16 ended, score: 1
Episode 17 ended, score: 1
Episode 18 ended, score: 1
Episode 19 ended, score: 3
Episode 20 ended, score: 1
Episode 21 ended, score: 2
Episode 22 ended, score: 0
Episode 23 ended, score: 2
Episode 24 ended, score: 2
Episode 25 ended, score: 0
Episode 26 ended, score: 0
Episode 27 ended, score: 1
Episode 28 ended, score: 2
Episode 29 ended, score: 3
Episode 30 ended, score: 2
Episode 31 ended, score: 0
Episode 32 ended, score: 6
Episode 33 ended, score: 2
Episode 34 ended, score: 1
Episode 35 ended, score: 2
Episode 36 ended, score: 1
Episode 37 ended, score: 2
Episode 38 ended, score: 2
Episode 39 ended, score: 3
Episode 40 ended, score: 0
Episode 41 ended, score: 0
```

Implimentation

Algorithm

- We have implemented the algorithm for Q-Learning
- We tried implementing CNNs on CPU and a Low End GPU and couldn't get satisfactory result.
- Now we will try to implement it on a better GPU.

Platform

- We are using Python implementation of ALE
- To use GPU for CNNs, Theano library of Python is used
- GPU available is Nvidia GTX 760
- Special Thanks to Prof. Vinay Namboodiri for making the GPU available

Refrences

[1] Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.

[2] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013)