Building an artificial mind that learns to play ATARI video games*

IT Akadeemia
toetab Skype™

Kristjan Korjus, Ilya Kuzovkin, Ardi Tampuu, Taivo Pungas[#]
Institute of Computer Science, University of Tartu, [#]tpungas@ut.ee

COMPUTATIONAL NEUROSCIENCE
Research group at the University of Tartu

Introduction

This project aims to reproduce a system, originally published by researchers at DeepMind Technologies, that is able to learn to play several ATARI video games using a single learning algorithm and in some cases, **perform better than a human expert**. The only inputs to the system are game screens and rewards.

*In this work, we follow as closely as possible the article: Mnih, V. et al. "**Playing Atari with Deep Reinforcement Learning.**" *arXiv preprint arXiv:1312.5602* (2013).

Task

The system receives a picture of an ATARI video game screen (an example is shown in Figure 1) and chooses an action to take.
It then executes this action and is told whether the score increased, decreased or did not change.

Based on this information and playing a large number of games, the system needs to learn to improve its performance in the game.

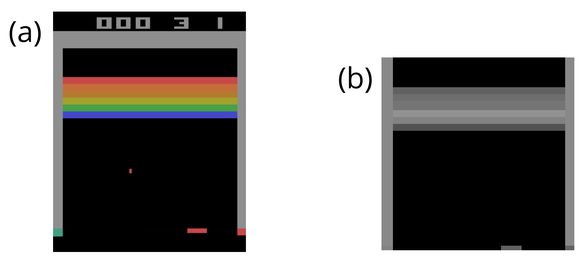

Figure 1. (a) A game screen of Breakout. (b) A processed game screen of Breakout.

Main components

Convolutional neural network

The neural network (shown in Figure 2) assigns an expected reward value to each possible action. Convolutional NN-s treat images as 2D objects and convolve them with linear filters to obtain activities of the next layer.

Q-learning

The concept of reinforcement learning is implemented as:

$$Q_{t+1}(s_t, a_t) = r_t + \gamma \cdot \max_a Q(s_{t+1}, a)$$

Here, $Q(s,a)$ is the predicted reward from taking action a while being in state s. r_t is the actual reward received after taking action a_t in state s_t.

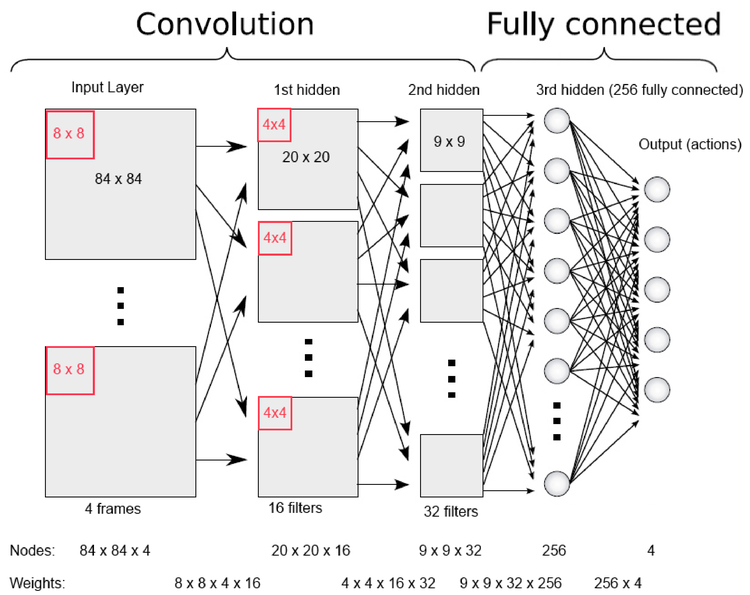

Figure 2. A diagram representing the organisation of the deep neural network used. The bottom two rows show the number of nodes and the number of independent weights.

Algorithm

```
Play many games
  Play many frames
    Get a frame and pre-process it (Fig. 1)
    Choose action using Deep Neural Net (Fig. 2) and current state
      Choose random action with probability 0.05
    Observe reward and save transition to memory
    Train the network
      32 random transitions: 32 x (state, reward, action, state)
      Q-learning: Reward = r_t + 0.9 · DNN(State_{t+1})
      Gradient descent
```

Technical information

The system is implemented using Python 2.7 with libraries `numpy` and `theano`.
The emulator used is Arcade Learning Environment (ALE).
Progress (Jun 2014): the system plays Breakout with randomly initialised neural network. Learning has not been confirmed.

Authors' positions at the Institute of Computer Science, Faculty of Mathematics and Computer Science, University of Tartu:

Kristjan Korjus	PhD student in Computer Science
Ilya Kuzovkin	PhD student in Computer Science
Ardi Tampuu	Programmer
Taivo Pungas	BSc student in Computer Science

Project website:
github.com/kristjankorjus/Replicating-DeepMind

UNIVERSITY OF TARTU
1632
Institute of Computer Science