# Building an Artificially Intelligent Machine

## Using Atari 2600 to Emphasize the Viability of Reinforcement Learning

An Integrated Research Component by Nathan Metze

Being Presented on Tuesday, November 14th 2017 at 5:30

**Abstract:** Within the field of machine learning, there exists a subfield known as reinforcement learning. In this paper, a Deep Q-learner model, a type of reinforcement learning, is selected to interpret high-dimensional sensory input with the goal of returning a value function that estimates future rewards, or actions to take. For this paper, a deep Q-learner model, or agent, is self-taught how to play an Atari 2600 video game, Enduro. This understanding of the game can be derived by the agent's ability to repeatedly perform positive actions over a given amount of time. This paper helps emphasize the viability of reinforcement learning using deep Q-learning in test environments such as the Atari 2600.

## Training

**Time Spent Training:**
- 10 days

**Frames Processed:**
- 11,115,632

**System Architecture:**
- The agent was trained using VMware
  - Operating System
    - Ubuntu 17.10
    - 12GB Ram
    - Intel I7-6700K CPU @ 4.0 GHz
  - Tools Used
    - OpenAI Gym
    - TensorFlow

## Methodology

The algorithms used to train the deep Q-learner model includes both a convolutional neural network and a Q-learner. In order for the Q-learner to effectively decide an action it must compute every possible state, which in a simple instance would be 110x110x4, for four images. Thus, the convolutional neural network is used to speed up this process by mapping all the possible states to the amount of action states available in the game. This significantly improves the computational speed.

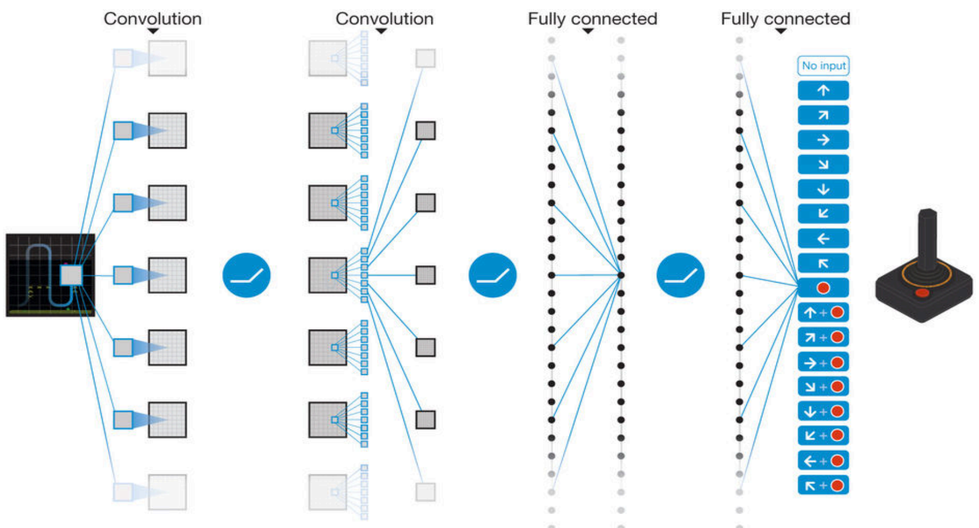


**Figure 1: Convolutional Neural Network with 5 Layers used to develop the deep q-learner[1]**

$$Q^*(s,a) = \max \sum \left[ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ \ldots \ \middle| s_t = s, a_t = a, \pi \right]$$

**Figure 2: Q-learner algorithm used - based upon the Bellman equation**

## Processing the Data

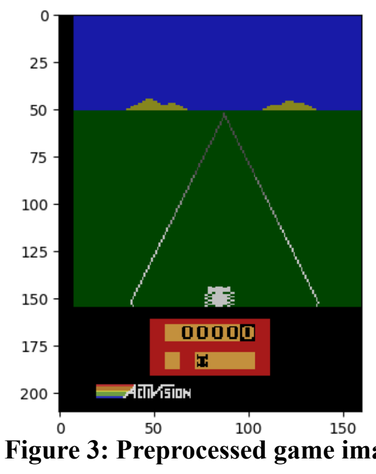

**Figure 3: Preprocessed game image**

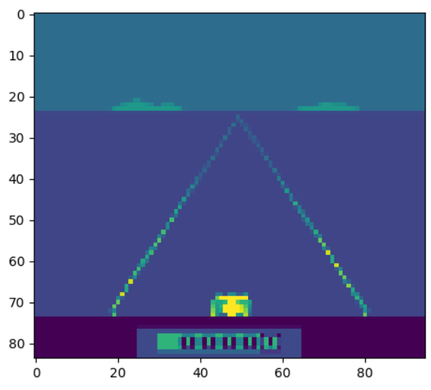

**Figure 4: Processed game image**

The initial image is processed to remove features including red, green, blue colors as well as the black border. This makes it easier for the neural network to find important features.

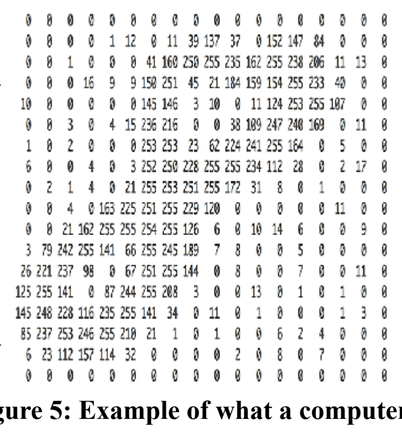

**Figure 5: Example of what a computer sees**

## Results

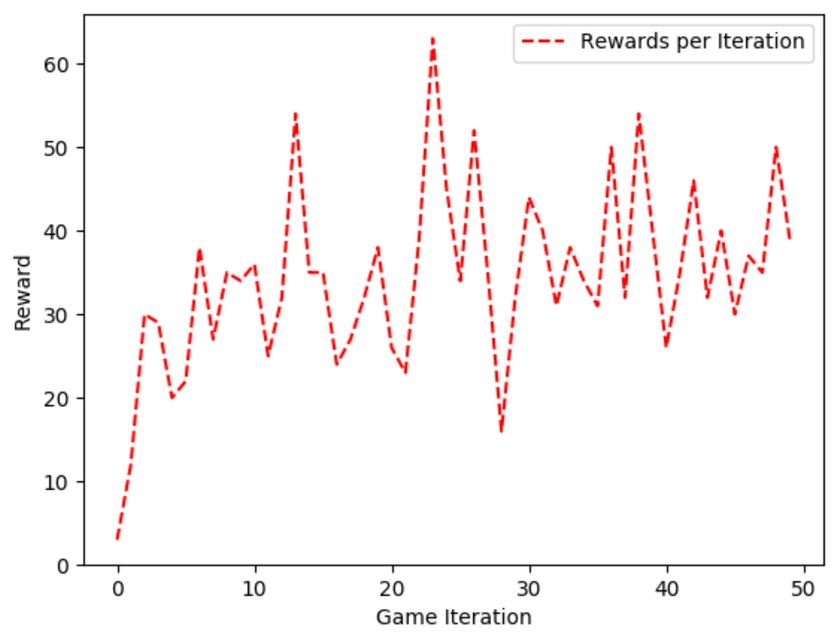


**Figure 6: Q-learner algorithm used - based upon the Bellman equation**

**Benchmarking:** The agent was evaluated by playing fifty iterations of the game while only taking an action every fourth frame. This was done to ensure the agent was on the same playing field as a human player, who can only realistically act on every fourth visible frame.

**Hyperparameters:**

gamma: 0.95 (reward discount rate)
epsilon: 1.00 (exploration rate)
epsilon decay: 0.99 (rate of decay)
max frames: 200,000 (frames saved)
mini batch size: 32 (sample size)

| Layer | Depth | Filter | Stride | Activation |
|---|---|---|---|---|
| Convolution-1 | 32 | 8x8 | 4 | ReLU |
| Convolution-2 | 64 | 4x4 | 2 | ReLU |
| Convolution-3 | 64 | 3x3 | 1 | ReLU |
| Connected-1 | 512 | N/A | N/A | ReLU |
| Connected-2 | # of actions (9) | N/A | N/A | Linear |

**Figure 7: Convolutional Neural Network Architecture**

1. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning" *Nature International Journal of Science*, 25 Feb. 2015, https://www.nature.com/articles/nature14236. Accessed 12 Sept. 2017.