Playing CHIP-8 Games with Reinforcement Learning

Niven Achenjang, Patrick DeMichele, Sam Rogers

What is CHIP-8?

Invented in the 1970s for programming video games on microcomputers, CHIP-8 is a programming language that supported many classic arcade games such as *Pong, Pac-Man,* and *Space Invaders*

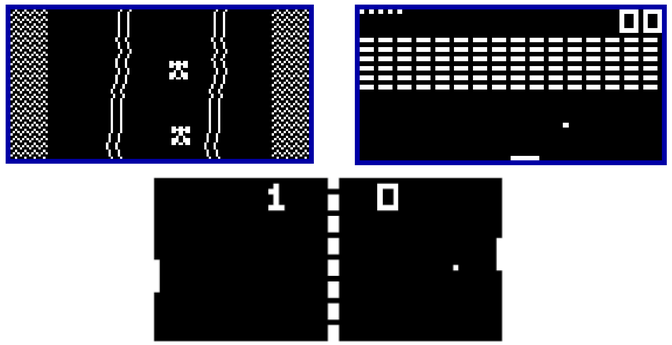


Fig 1. Some CHIP-8 Games

Background and Motivation

In 2008, DeepMind Technologies released a groundbreaking paper in which they used a method of reinforcement learning known as Deep Q-Learning to play seven games for the Atari 2600. This paper presented “the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning.”

Our aim was to replicate some of DeepMind’s results in a scaled-back setting: CHIP-8. While the data used in the DeepMind experiment were large and in full color (screens were 84 x 84 x 4 float arrays), CHIP-8’s smaller, monochrome screen allowed us to use data of a much smaller dimension (32 x 64 boolean arrays). Moreover, the use of a general CHIP-8 emulator allows our implementation to be adapted to multiple CHIP-8 games.

Deep Q-learning

What is Q-Learning?

Q-learning is a reinforcement learning technique used to find an optimal action-selection policy for a given Markov decision process. It seeks to learn the expected utility (the “quality”) of each state-action pair via feedback from experience.

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Fig 2. Update Rule for Q-Learning

Q-learning can be compared to value iteration, with values stored for state-action pairs rather than states. While Q-learning may require significant memory to store values for all state-action pairs, it also saves memory in that it does not require explicit estimation of transition probabilities.

What is Deep Q-Learning?

For an RL task where states are images, storing values for each state-action pair explicitly is impossible, and discretization may not generalize well. A solution to this is to approximate Q by a neural network which takes a state vector as input and has output nodes with corresponding values for each action. Then, the update rule is possible by standard gradient descent.

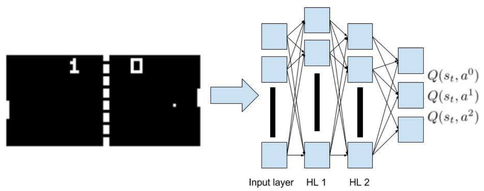


Fig 3. A Deep Q-Learning Network

Methodology

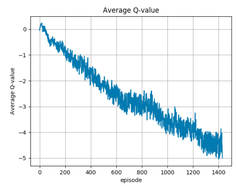
In addition to utilizing the Deep Q-Learning model outlined above, we experimented with several different parameters for the model, including

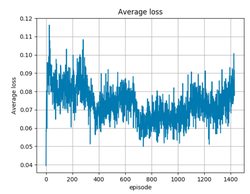
- Using replay memory for updating the model, in which transitions are stored in a memory bank and sampled whenever backpropagation is performed. This allows for better stability than sequential updates.
- Implementing epsilon-greedy exploration, where the model “explores” randomly at the beginning of the experiment and slowly transfers over to more network-computed actions.
- L2 regularization of network parameters
- The addition of Gaussian noise to the screen states

Results

Here we present results from two of our experiments.

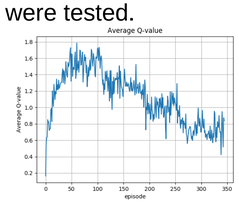
In the first experiment we used a standard feedforward neural network, which gave very poor results as evidenced in the following two graphs

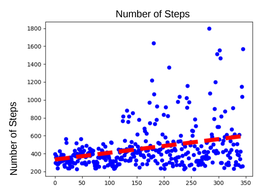




Here we see the best Q-Value for a given state quickly become negative and decrease with the number of episodes (games). Moreover the average loss falls initially, before increasing after about 700 episodes.

In our second test we used a convolutional network which gave slightly more promising results, although due to computation times fewer episodes were tested.





Here the Q-values remain positive throughout testing. In the second graph, see the average number of steps (frames) per game increases, showing that the agent is more successful at returning the ball as episodes increase.

References

Fig 1: http://www.pong-story.com/chip8/car.gif
http://www.pong-story.com/chip8/brix.gif

DeepMind Technologies: “Playing Atari with Deep Reinforcement Learning,” arXiv:1312.5602