Playing Atari With Deep Reinforcement Learning
This AI is Special

• No access to the emulator internals.
  • Doesn’t know everything.

• Limited hand-holding.
  • Doesn’t need a specially crafted feature representation.

• In general, just leave it alone.
  • With sufficient data, it is often possible to learn better representations than handcrafted features.
Preprocessing

- Gray-scale
  - RGB (128, 128, 128)

- Down-sampling
  - 210x160 -> 110x84

- Crop
  - 84x84
**Numbers**

- 10 million frames
- Frame skipping (selecting every 3\textsuperscript{rd} or 4\textsuperscript{th} frame)

- \(10000000 \text{ frames} \times \frac{4 \text{ actual frames}}{60 \text{Hz}}\)
- 7.71604938271605 days
- 185.1851851851852 hours
Reinforcement Learning

• Use samples to optimize performance.
• Use function approximation to capture large environments.
  • Appropriate methods for representing the value function

• Model-based vs Model-free learning

• On-policy vs Off-policy
On Off-policy and On-policy

- Q-learning

\[
Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) \times \left[ R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right]
\]

- SARSA

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\]
Deep Q-learning

- Part of the family of fitted value iteration algorithms using ‘experience replay’
  - Comparable to Neural fitted Q-learning (NFQ)
    - Different parameter update (RPROP vs RMSProp)
    - Batch update difference (All vs stochastic among last 1 million)
  - Benefits
    - Data efficiency (Samples may be reused)
    - Update variance reduction (Randomizing examples is good)
    - Smooths out learning (Off-policy -> behavior distribution)
  - Must learn off-policy
    - Current parameters probably not the same as the replayed sample
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights

for episode = 1, $M$ do
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ do
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} 
      r_j & \text{for terminal } \phi_{j+1} \\
      r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1}
    \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for
Convolutional Neural Networks

- The exception to deep supervised neural networks!
- Inspired by the visual system’s structure.
- Applying the same neurons to different patches yields a form of translational invariance.
Convolutional Neural Networks

- Input: 84x84x4
- Convolutional Layers:
  - L1: 8x8
  - L2: 4x4
  - L3:
  - L4:
- Fully Connected Layers:
  - F5: 256
  - F6: 4-18 (Output)

Stole this page
Hinton's work centers around not needing to find good features
- He argues that once you have the right features from the data, the algorithm you pick is relatively unimportant
- The normal process is very intuitive and requires significant hands on work by AI developers
- Other approaches try to automatically determine the “best” features before passing them to the classifier, but often at a significant computational cost
- The goal is then to find algorithms (both training and architecturally) to not explicitly do that feature discovery work, but to build a system directly from the data itself
Training and Stability

Average Reward on Breakout

Average Reward on Seaquest
Training and Stability

Average Q on Breakout

Average Action Value (Q)

Training Epochs

Average Q on Seaquest

Average Action Value (Q)

Training Epochs
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>B. Rider</th>
<th>Breakout</th>
<th>Enduro</th>
<th>Pong</th>
<th>Q*bert</th>
<th>Seaquest</th>
<th>S. Invaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>354</td>
<td>1.2</td>
<td>0</td>
<td>−20.4</td>
<td>157</td>
<td>110</td>
<td>179</td>
</tr>
<tr>
<td>Contingency [4]</td>
<td>1743</td>
<td>6</td>
<td>159</td>
<td>−17</td>
<td>960</td>
<td>723</td>
<td>268</td>
</tr>
<tr>
<td>DQN</td>
<td><strong>4092</strong></td>
<td><strong>168</strong></td>
<td><strong>470</strong></td>
<td>20</td>
<td><strong>1952</strong></td>
<td><strong>1705</strong></td>
<td><strong>581</strong></td>
</tr>
<tr>
<td>Human</td>
<td>7456</td>
<td>31</td>
<td>368</td>
<td>−3</td>
<td>18900</td>
<td>28010</td>
<td>3690</td>
</tr>
<tr>
<td>HNet Best [8]</td>
<td>3616</td>
<td>52</td>
<td>106</td>
<td>19</td>
<td>1800</td>
<td>920</td>
<td><strong>1720</strong></td>
</tr>
<tr>
<td>HNet Pixel [8]</td>
<td>1332</td>
<td>4</td>
<td>91</td>
<td>−16</td>
<td>1325</td>
<td>800</td>
<td>1145</td>
</tr>
<tr>
<td>DQN Best</td>
<td><strong>5184</strong></td>
<td><strong>225</strong></td>
<td><strong>661</strong></td>
<td>21</td>
<td><strong>4500</strong></td>
<td><strong>1740</strong></td>
<td>1075</td>
</tr>
</tbody>
</table>