Playing Atari with Deep Reinforcement Learning

Jonathan Chung
Objectives

• Provide some basic understanding of RL
• Apply this understanding to the paper
• Discuss possible future directions of the paper
Reinforcement Learning

“Finding suitable actions in order to maximise the reward”
Problem definition

- States (st)
- Action (a)
- Reward (r)
Problem definition

Markov decision process
Problem definition

• States (st)
  Where you are (and where you have been)

• Action (a)
  What can you do

• Reward (r)
  What can you get
Rewards are not always immediate.
Problem definition

- **States (st)**
  Where you are (and where you have been)

- **Action (a)**
  What can you do

- **Reward (r)**
  What can you get
What happens in the past affects what happens in the future.
Problem definition

- **States (st)**
  Where you are (and where you have been)

- **Action (a)**
  What can you do

- **Reward (r)**
  What can you get
Intelligence is the ability to adapt to change.
Problem definition

- **States (st)**
  Where you are (and where you have been)

- **Action (a)**
  What can you do

- **Reward (r)**
  What can you get
Aim

\[ Q^*(s, a) = \max_\pi \mathbb{E} [R_t | s_t = s, a_t = a, \pi] \]

- \( Q \) is defined as the maximum expected reward \( (R_t) \) after sequence \( s \) and taking action \( a \).
Playing Atari with Deep Reinforcement Learning


\[ Q^*(s, a) = \max_{\pi} \mathbb{E} [R_t | s_t = s, a_t = a, \pi] \]

- Images \sim s^#
- Actions = a^\wedge
- Score \sim Reward^*

#A sequence of images are dependent on the sequence s.
**Inputs**

\[ Q^*(s, a) = \max_\pi \mathbb{E}[R_t | s_t = s, a_t = a, \pi] \]

- Images \( \sim s^# \)
- Actions \( = a^\wedge \)
- Score \( \sim \text{Reward}^* \)

*All future rewards are considered but discounted based on the time*
Learning

- Bellman equation

\[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \mid s, a \right] \]
Learning

• Bellman equation

\[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \mid s, a \right] \]
Learning

• Bellman equation

\[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \bigg| s, a \right] \]

• s - sequence, a series of states

• a - action
Learning

- Bellman equation

\[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \mid s, a \right] \]
Learning

- Bellman equation

\[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \middle| s, a \right] \]
Learning

- Bellman equation

\[
Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \middle| s, a \right]
\]

Discount factor $\gamma$

Maximum expected reward $R_t$

$st_0 \rightarrow st_1 \rightarrow st_2 \rightarrow st_3 \rightarrow st_4 \rightarrow st_5 \rightarrow st_6 \rightarrow st_7$

$r = 1 \rightarrow r = 3 \rightarrow r = 4 \rightarrow r = 6 \rightarrow r = 7$

$t = t'$
Learning

- Bellman equation

\[
Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \mid s, a \right]
\]
Learning

- Bellman equation
  \[ Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a') \mid s, a \right] \]

- \( Q_i \to Q^* \) as \( i \to \infty \)
- \( Q(s, a; \theta) \approx Q^*(s, a) \)
- where \( Q(s, a; \theta) \) is modelled with a deep neural network called a “Q-network”
$Q(s, a, \theta)$

Loss func. MSE

Down sample & crop

Fully connected (256 units)

CNN(32 4x4 filters, stride = 2)

CNN(16 8x8 filters, stride = 2)

$st_0 \rightarrow a_i \rightarrow st_1 \rightarrow a_i \rightarrow st_2 \rightarrow a \rightarrow st_3 \rightarrow a \rightarrow st_4 \rightarrow a_i \rightarrow st_5 \rightarrow a_i \rightarrow st_6 \rightarrow a_i \rightarrow st_7$

$r = 1$

$r = 3$

$r = 4$

$r = 5$

$r = 6$

$r = 7$

$r + \gamma Q(s', a', \theta)$
Deep Q-learning

- Initialise data and Q weights
- For each episode:
  - Init. and preprocess sequence $\phi(s_t)$
  - For $t$ in $T$
    - Select the best action $a_t$ according to $Q(\phi(s_t), a, \theta)$
    - Execute action to get reward $r_t$ and image $x_{t+1}$
    - Store $\phi(s_t), a_t, r_t, \phi(s_{t+1}) \rightarrow D$
    - Sample a mini batch $\hat{\omega}$ from $D$ then perform gradient descent to update weights
Experiments

- 7 ATARI game (Beam rider, Breakout, Enduro, Pong, Q*bert, Seaquest, Space Invaders)

- Each trained on the same network (except actions, and scaled rewards according)

- Sequences contained the actions and states from the last 4 frames
Experiments

- Randomly sampled s, a and s’, a’
- RMSProp algorithm with mini batch of size 32
- Total of 10 million frames while training on every 4 (or 3) frames
Experiments

- Outperformed previous state of the art Sarsa and Contingency
- Performed better than humans in Breakout, Enduro, Pong

<table>
<thead>
<tr>
<th></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>4092</td>
<td>168</td>
<td>470</td>
<td>20</td>
<td>1952</td>
<td>1705</td>
<td>581</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>HNeat Best [8]</td>
<td>3616</td>
<td>52</td>
<td>106</td>
<td>19</td>
<td>1800</td>
<td>920</td>
<td>1720</td>
</tr>
<tr>
<td>HNeat 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>5184</td>
<td>225</td>
<td>661</td>
<td>21</td>
<td>4500</td>
<td>1740</td>
<td>1075</td>
</tr>
</tbody>
</table>
Future direction
Future direction

units killed = 30
territory = 30%

units killed = 50
territory = 33%

WIN

st_0 \rightarrow a_i \rightarrow st_1 \rightarrow a_i \rightarrow st_2 \rightarrow a_i \rightarrow st_3 \rightarrow a_i \rightarrow st_4 \rightarrow a_i \rightarrow st_5 \rightarrow a_i \rightarrow st_6 \rightarrow a_i \rightarrow st_7