*Playing Atari with Deep Reinforcement Learning
*Human-Level Control Through Deep Reinforcement Learning
†Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

*Mnih et al., Google Deepmind
† Guo et al., University of Michigan

Reviewed by Zhao Song

April 10, 2015
1 Introduction

2 Deep Q-network

3 Monte Carlo Tree Search Planning
Outline

1. Introduction

2. Deep Q-network

3. Monte Carlo Tree Search Planning
Markov decision process (MDP)

Figure 1: An MDP model illustration from [Kaelbling et al. 1998].

- Described as a tuple \(<S, A, T, R>\) where
  - \(S\) is a finite set of states of the world.
  - \(A\) is a finite set of actions.
  - \(T: S \times A \rightarrow \pi(S)\) denotes the state transition function.
  - \(R: S \times A \rightarrow \mathbb{R}\) corresponds to the reward function.
- Policy: \(S \rightarrow A\).
- Goal: Maximizing the expected reward \(\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]\).
Partially observable Markov decision process (POMDP)

Figure 2: A POMDP model illustration from [Kaelbling et al. 1998].

- State not observable.
- In addition to the tuple $< S, A, T, R >$, we need to define
  - $\Omega$: a finite set of observations.
  - $O : S \times A \rightarrow \pi(\Omega)$ denotes the observation function.
- The belief state $b(s)$.
- Policy: $B \rightarrow A$. 
Reinforcement learning (RL)

Model unknown.
 State usually invisible to the agent.
 At each time \( t \), the agent
  - executes an action \( a_t \)
  - receives an observation \( o_t \) (or state \( s_t \))
  - receives a reward \( r_t \)

Goal: Learning policy from history \( h_t = \{ a_0, o_1, r_1, \ldots, a_{t-1}, o_t, r_t \} \).
RL example: the Atari 2600 game

Figure 4: Snapshots from five Atari games shown in [Mnih, Kavukcuoglu, Silver, Graves, et al. 2013]
RL example: the Atari 2600 game (Cont.)

Figure 5: RL model for Atari games shown in [Silver 2015]
Outline

1. Introduction
2. Deep Q-network
3. Monte Carlo Tree Search Planning
Overview of the deep Q-network (DQN)

- First approach to combine RL and deep learning (DL).
  - Q-learning to update the state-action function value.
  - DL to train a convolutional neural network (CNN).
- Covert the POMDP model into the MDP model.
  - Treating the 4 most recent frames as the input state.
- Superior performances on some games, compared with human experts.
Q-learning in DQN

- Updating the action-value function $Q$ to approximate the true value function $V$.
- Bellman update:

$$Q_{i+1}(s, a) = \mathbb{E}_{s' \sim \epsilon}[r + \gamma \max_{a'} Q_i(s', a') | s, a]$$

- Function approximation $Q(s, a; \theta) \approx Q^*(s, a)$.
- Model free.
The neural network scheme

Figure 6: The CNN scheme shown in [Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015]
The proposed algorithm

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, $M$ do
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1, T$ do
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
    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 $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  End For
End For

Figure 7: The algorithm of deep Q-learning with experience replay shown in [Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015]
Implementation details

- Stochastic gradient descent to learn the neural network.
- Rescale the original $210 \times 160$ RGB images to $84 \times 84$ gray scale images.
- Last 4 frames as input to DQN.
- Set positive rewards to 1 and negative rewards to $-1$.
- 50,000,000 frames in training (around 38 days of game experience).
- $\epsilon$-greedy policy:
  - Annealed linearly from 1.0 to 0.1 over the first one million frames and then 0.1 thereafter.
Results from [Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015]
Outline

1 Introduction

2 Deep Q-network

3 Monte Carlo Tree Search Planning
Motivation

- State of each game available in RAM, although not observed by human player.
- Solve an MDP problem by the upper confidence bound for tree (UCT) method:
  - Orders of magnitude slower than the RL agent.
  - Higher policy values.
- How to combine the planning agent and the RL agent?
Overview of UCT planning agent

- Use the emulator as a model to simulate trajectories.
- How to decide the value of taking action at current state of the game?
- For state-depth pair \((s, d)\) in the \(k\)th trajectory, compute score for each action \(a\) according to the sum of
  - An exploitation term: Monte Carlo average of the discounted sum of rewards for state-depth pair \((s, d)\) in the previous \(k - 1\) trajectories.
  - An exploration term: \(\sqrt{\frac{\log(n(s, d))}{n(s, a, d)}}\)
- At the end of trajectories, return the exploitation term at the root node as the desired output.
- For most of the games, use 300 as the maximum depth and 10,000 as the number of trajectories.
The proposed agents

UCTtoRegression
- Play a game 800 times via UCT.
- Map the last four frames of each state into action value to obtain training data.
- Train the CNN via regression.
- Linear function applied to the output layer.

UCTtoClassification
- Play a game 800 times via UCT.
- Map the last four frames of each state into action choice to obtain training data.
- Train the CNN via multinomial classification.
- Softmax function applied to the output layer.
The proposed agents (Cont.)

UCTtoClassification-Interleaved

- Play a game 200 times via UCT.
- Train the CNN via multinomial classification.
- Play the game 200 times via the CNN trained.
- Iterate until 800 runs in total.
- Train the CNN via multinomial classification.
### Table 1: Performance (game scores) of the four real-time game playing agents, where UCR is short for UCT-toRegression, UCC is short for UCT-toClassification, and UCC-I is short for UCT-toClassification-Interleaved.

<table>
<thead>
<tr>
<th>Agent</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>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></td>
<td>-best</td>
<td>5184</td>
<td>225</td>
<td>661</td>
<td>4500</td>
<td>1740</td>
<td>1075</td>
</tr>
<tr>
<td>UCC</td>
<td>5342(20)</td>
<td>175(5.63)</td>
<td>558(14)</td>
<td>19(0.3)</td>
<td>11574(44)</td>
<td>2273(23)</td>
<td>672(5.3)</td>
</tr>
<tr>
<td></td>
<td>-best</td>
<td>10514</td>
<td>351</td>
<td>942</td>
<td>29725</td>
<td>5100</td>
<td>1200</td>
</tr>
<tr>
<td></td>
<td>-greedy</td>
<td>5676</td>
<td>269</td>
<td>692</td>
<td>19890</td>
<td>2760</td>
<td>680</td>
</tr>
<tr>
<td>UCC-I</td>
<td>5388(4.6)</td>
<td>215(6.69)</td>
<td>601(11)</td>
<td>19(0.14)</td>
<td>13189(35.3)</td>
<td>2701(6.09)</td>
<td>670(4.24)</td>
</tr>
<tr>
<td></td>
<td>-best</td>
<td>10732</td>
<td>413</td>
<td>1026</td>
<td>29900</td>
<td>6100</td>
<td>910</td>
</tr>
<tr>
<td></td>
<td>-greedy</td>
<td>5702</td>
<td>380</td>
<td>741</td>
<td>20025</td>
<td>2995</td>
<td>692</td>
</tr>
<tr>
<td>UCR</td>
<td>2405(12)</td>
<td>143(6.7)</td>
<td>566(10.2)</td>
<td>19(0.3)</td>
<td>12755(40.7)</td>
<td>1024(13.8)</td>
<td>441(8.1)</td>
</tr>
</tbody>
</table>

### Table 2: Performance (game scores) of the off-line UCT game playing agent.

<table>
<thead>
<tr>
<th>Agent</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>UCT</td>
<td>7233</td>
<td>406</td>
<td>788</td>
<td>21</td>
<td>18850</td>
<td>3257</td>
<td>2354</td>
</tr>
</tbody>
</table>

**Figure 8**: The comparison of policy values shown in [Guo et al. 2014]

