AIL 722
Case Studies with DQN in Real Life

Ref: S&B Chapter 16

Vaibhav Bihani & Sanket Gandhi

Challenges in applying RL in real life

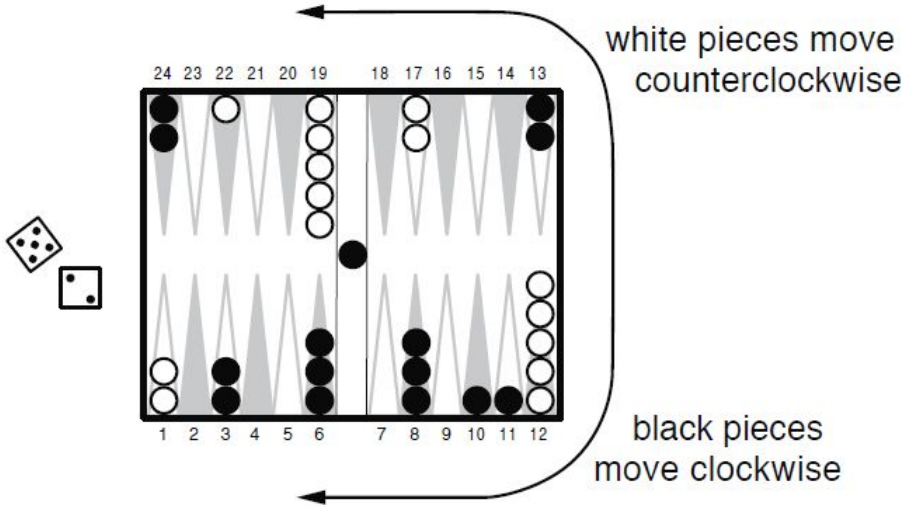
- How to represent/store value functions/policy?

  E.g. Backgammon : 30 pieces, 24 possible locations ($30^{24} \sim 10^{4}$ Zettabyte)

- How to represent state?
  - Feature designing
- Non-IID Data
- Training Instability

TD Gammon (1995,2002)

- Goal is to move all your checkers off the board before your opponent does.
- Roll two dice to determine points to move
- Hit : If lands on opponent's single checker , Removed and re-enters
- Cannot stack over opponent's stack



A backgammon position

TD-Gammon (1995,2002)

- Used a nonlinear form of TD(λ)
- Value function as single layer ANN
- After playing about 300,000 games against itself, reached level of best programmes at that time
- Learned to play certain opening positions differently than was the convention among the best human players

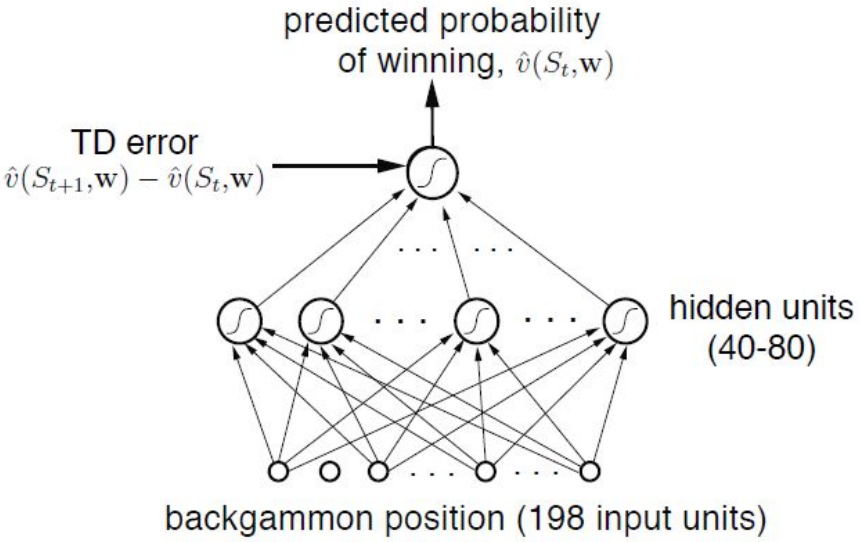
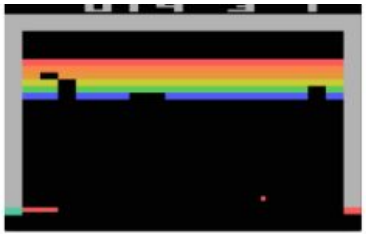


Figure 16.1: The TD-Gammon ANN

Atari Games

- Atari2600 Games : Arcade video games, first started in 1970s

Pong



Breakout

Space Invaders

Seaquest

Beam Rider

Atari Games

- Playing Atari with Deep Reinforcement Learning
- Minh et Al. (2013) at DeepMind
- Shown that same architecture can learn most of the 49 Atari games at a level of humans
- The first step towards “General Artificial Intelligence”
- Google acquired DeepMind in 2014

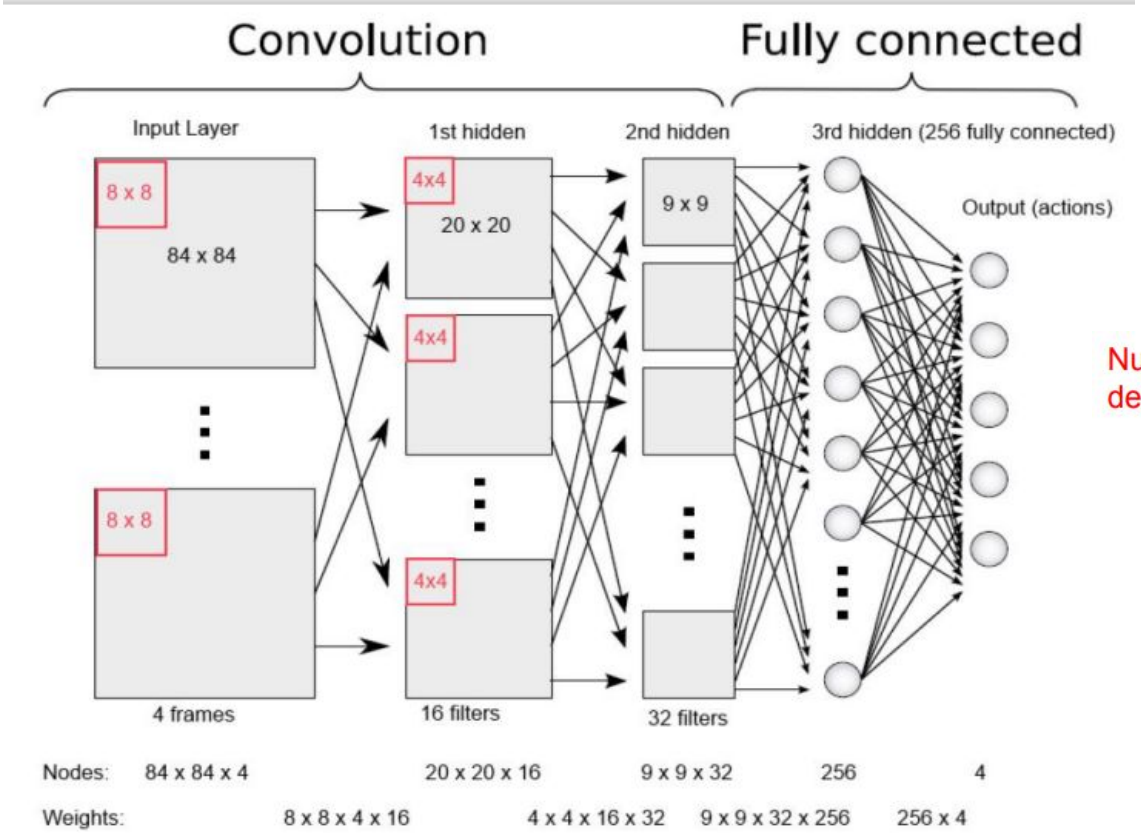
Architecture : Preprocessing

1) Original 210 x 160 with 128 color palette (High Dim , Large Mem Req,)
2) Cropping , Rescaling, Downsample
3) Take Max value over current and last frames for each pixel color value (Removes flickering of game)
4) Take Y channel (Luminescence) and rescale to 1
5) Final Image 84 x 84 x 1
6) Stack last 4
7) 84 x 84 x 4

- State: screen pixels
  - Image size: 84×84 (resized)
  - Consecutive **4** images
  - Grayscale with **256** gray levels

$256^{84 \times 84 \times 4}$ rows in the Q-table!

Architecture : ConvNet



Number of actions between 4-18 depending on Atari game

Training : Loss Function

- Loss Function:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

- Gradient Update

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

Training : Issues

- Naïve Q-learning oscillates or diverges with neural nets
    - Data is sequential :Successive samples are correlated, non-i.i.d.
    - Policy changes rapidly with slight changes to Q-values Policy may oscillate. Distribution of data can swing from one extreme to another
    - Scale of rewards and Q-values is unknown : Gradients can be large unstable when backpropagated

Training : Experience Replay

- Learning from batches of consecutive samples is problematic: -
  - Samples are correlated => inefficient learning
- Address these problems using experience replay
  - Continually update a replay memory table of transitions (st , at , rt , st+1) as game (experience) episodes are played
  - Train Q-network on random minibatches of transitions from the replay memory, instead of consecutive samples

Training : Freeze Target Q-Network and Reward Clipping

- Compute Q-learning targets w.r.t. old, fixed parameters θ_i^-

$$r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$$

- Optimize MSE between Q-network and Q-learning targets

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

- Periodically update fixed parameters $\theta_i^- \leftarrow \theta_i$

- Clip Rewards to [-1 ,1]
- Prevents large Q-values
- Ensures Well conditioned gradients

Training : Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

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 ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    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 $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

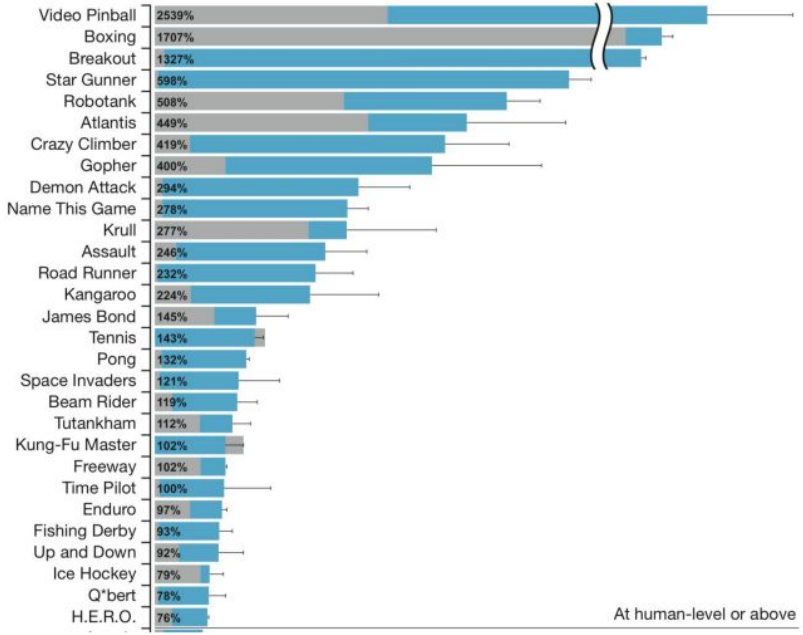
    Every C steps reset $\hat{Q} = Q$

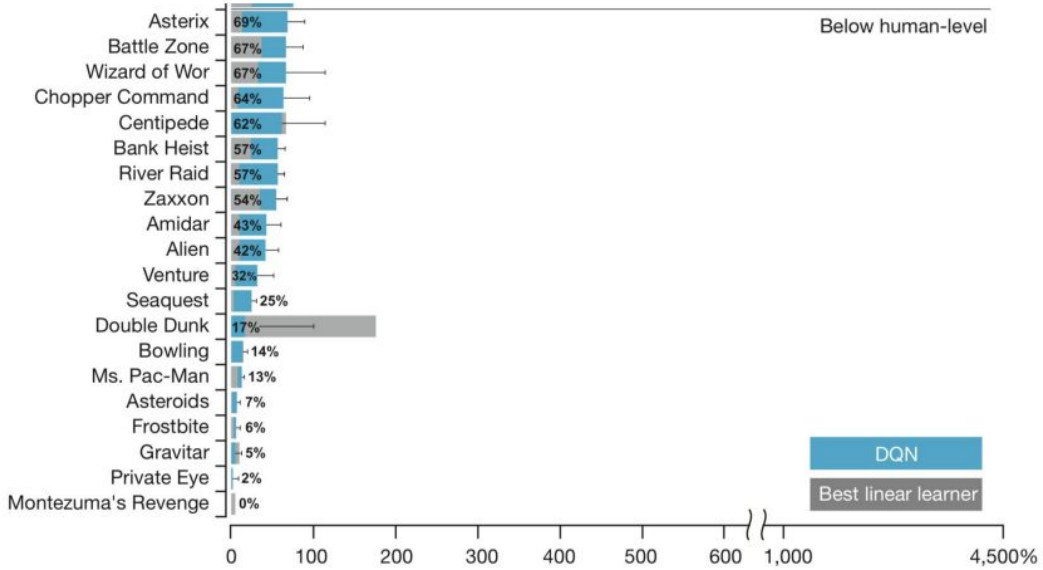
  End For

End For

Performance in Breakout Game

Results





100 * (DQN score - random play score)/ (human score - random play score)

Results

Extended Data Table 3 | The effects of replay and separating the target Q-network

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

DQN agents were trained for 10 million frames using standard hyperparameters for all possible combinations of turning replay on or off, using or not using a separate target Q-network, and three different learning rates. Each agent was evaluated every 250,000 training frames for 135,000 validation frames and the highest average episode score is reported. Note that these evaluation episodes were not truncated at 5 min leading to higher scores on Enduro than the ones reported in Extended Data Table 2. Note also that the number of training frames was shorter (10 million frames) as compared to the main results presented in Extended Data Table 2 (50 million frames).

Thank You