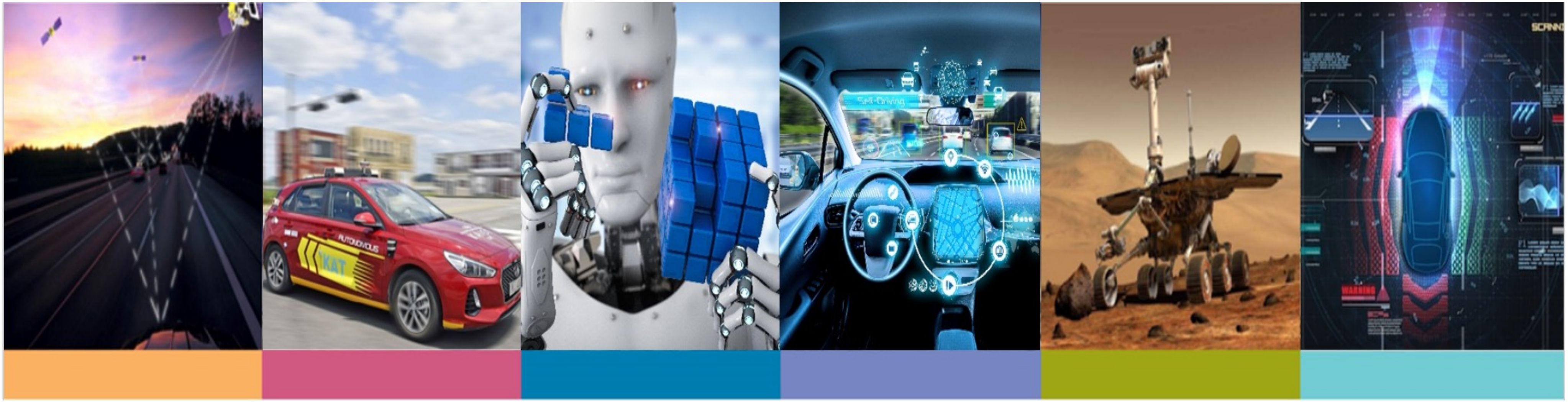


# Deep Q-Learning Network

## “Playing atari with deep reinforcement learning”
## “Human-level control through deep reinforcement learning”

2nd Seminar, 2023 AVE Lab Summer Internship
Hongtae Kim

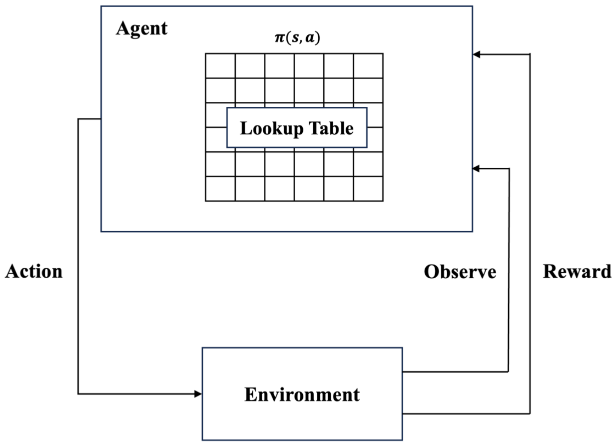
2023.08.11

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).
Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

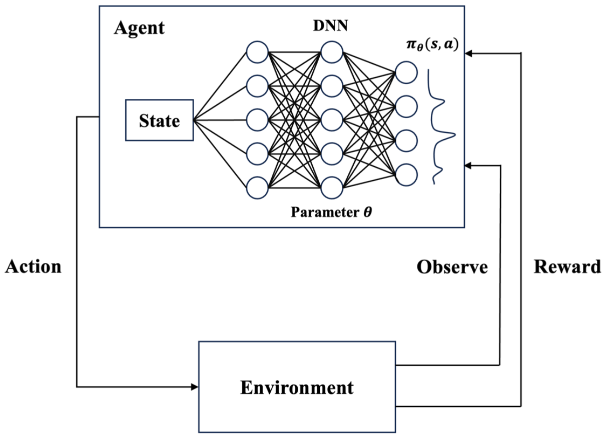
# Contents

# Deep Reinforcement Learning (Deep RL)

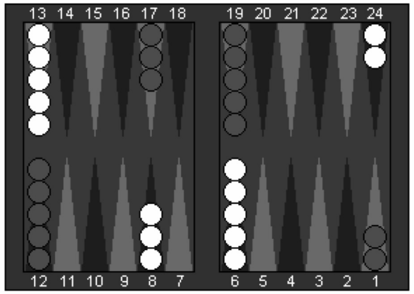
- Combine Deep Learning + Reinforcement Learning (Deep RL)
- Previously, RL with Tabular methods couldn't solve with large-scale state (practical), so RL with function approximation methods proposed
- TD-gammon was the first to combine DL and RL, but it did not consider convergence (Hand-craft feature, Correlation, Target move)




Tabular Method Reinforcement Learning




Deep Reinforcement Learning

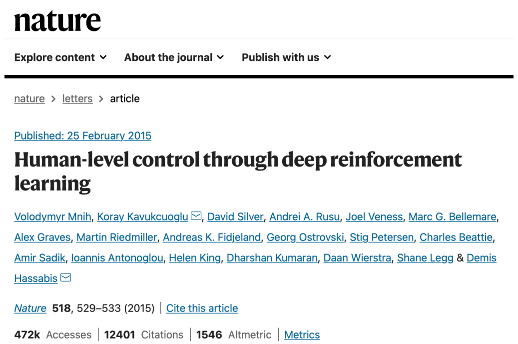



Backgammon

Tesauro, Gerald. "Temporal difference learning and TD-Gammon." *Communications of the ACM* 38.3 (1995): 58-68.

# Deep Q-Network (DQN)

- Published 2013 in arXiv, 2015 in Nature
- Only RL had limitations
    1. Useful features can be handcrafted
    2. Low-dimensional state space
- Combined DL (CNN) and RL (Q-learning), proposed a novel ‘Deep Q-Network’
- Possible to train using high-dimensional sensory input (like human)
- Applied to an Atari 2600, it outperforms a professional human expert in 29 out of 49 games


nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > letters > article

Published: 25 February 2015

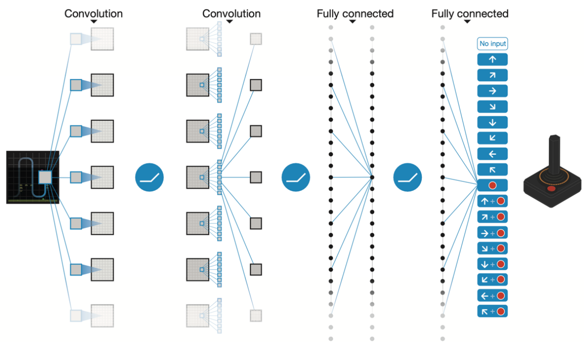
**Human-level control through deep reinforcement learning**

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

*Nature* **518**, 529–533 (2015) | Cite this article

**472k** Accesses | **12401** Citations | **1546** Altmetric | Metrics

Nature 2015




DQN Architecture

Atari 2600

# Q-Learning

- Q-learning is a model-free, off-policy algorithm
- Goal : Learn an optimal Q-function, $Q^*(s, a)$, so that the agent can obtain the maximum discounted cumulative reward from the environment
- While learning, use behavior policy to select action, and target policy for update Q-value
- Behavior policy : $\varepsilon$-greedy, Target policy : greedy

$$Action = \begin{cases} \max_{a \in A} Q(s_{t+1}, a) \ , & 1 - \varepsilon \\ random\ a \ , & \varepsilon \end{cases}$$

$\varepsilon$-greedy

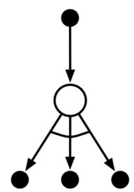
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \boldsymbol{\max_{a} Q(S_{t+1}, a)} - Q(S_t, A_t)\right]$$

greedy

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize Q(s, a), for all $s \in S^+, a \in A(s)$, arbitrarily except that Q(*terminal*, ·) = 0

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose A form S using policy derived from Q ($\varepsilon$-greedy)
    Take action A, observe R, $S'$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t)\right]$
    $S \leftarrow S'$
  until S is terminal

Pseudo-code for Q-learning



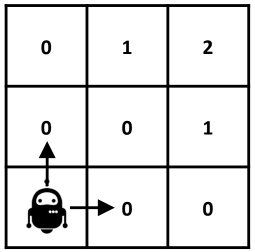
Q-learning

Backup diagrams for Q-learning

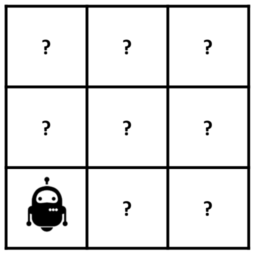
Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

# Model & Policy

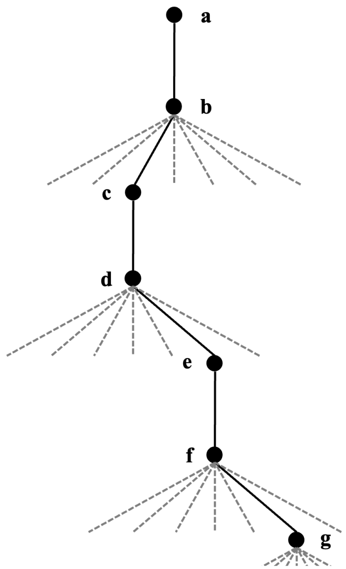
- Model-based : Model of the environment is known, Planning, Expected update
- Model-free : Model of the environment is unknown, Learning, Sample update
- On-policy : Single policy, No exploration after convergence
- Off-policy : Target policy & Behavior policy, Balance exploration and exploitation




Model-based




Model-free




On-policy & Off-policy

- Online Banner Advertisements
  Exploitation : Show the most successful advert
  Exploration : Show a different advert
- Game Playing
  Exploitation : Play the move you believe is best
  Exploration : Play an experimental move

Exploitation & Exploration

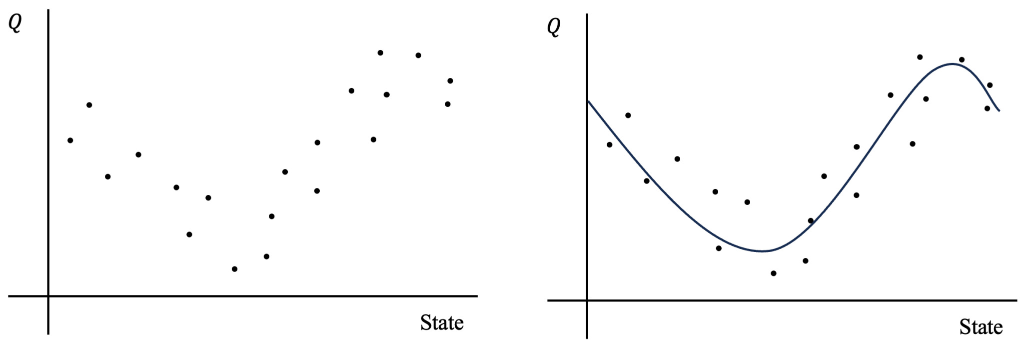
Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

# Function Approximation

- Q-learning is a method of storing Q values for all state-action pair in a table and updating it to find the optimal policy, called tabular method
- Large state spaces is not only memory (High-dimensional data) and computation, but also encountering states we have never seen before
- Generalization
- Function parameterized for a new variable w (weight)
- Function that approximates the true value using this parameter is called a function approximator

$$Q^*(s,a) \approx \hat{Q}(s,a,\boldsymbol{w})$$

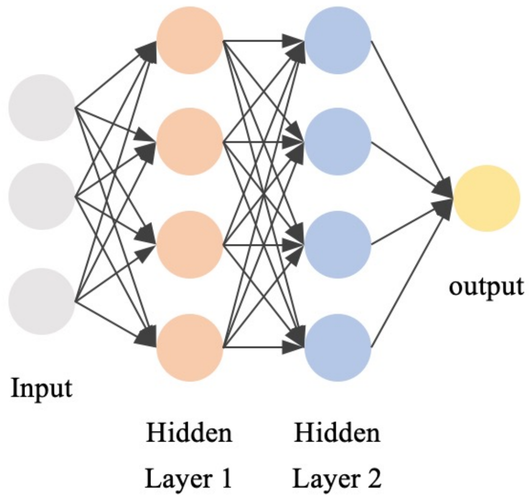
Function Approximator




Function Approximation

# Deep Learning

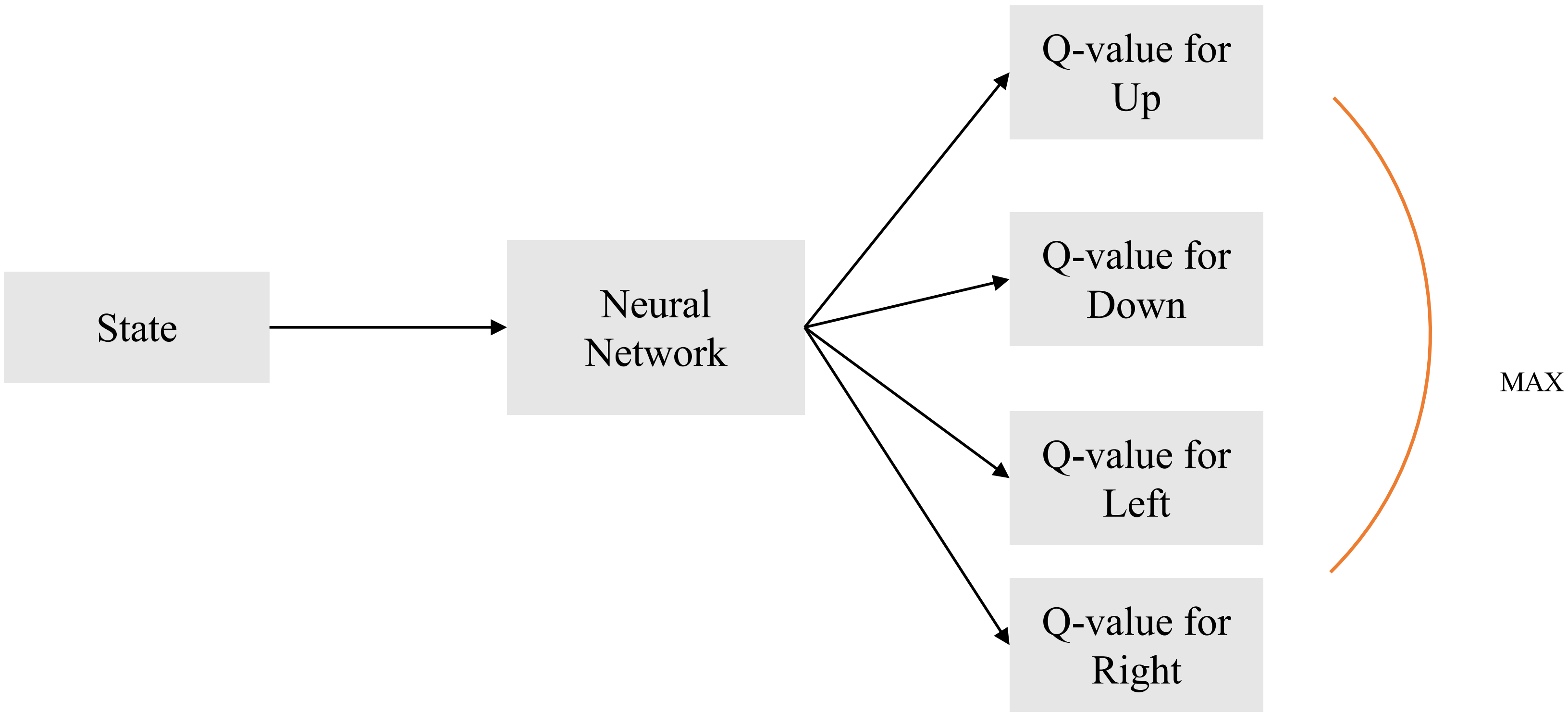
- In DQN, Deep learning is used to approximator
- To determine the relationship between X and y, find the weights
- Where training data and label are given (X : data, y : labels)
- Find $W_1, b_1, W_2, b_2, W_o, b_o$
- Function Approximation




$$y = f(\,a(\,a(\,a(X \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_o + b_o)\,)$$

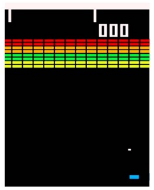
# DQN

- DQN = Deep learning + Q-learning
- Parameterizing Q function
- Forward pass

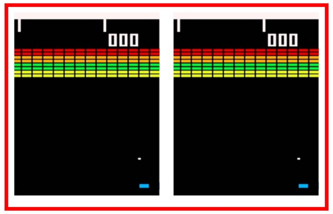


# Preprocessing

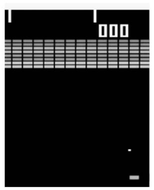
1. $210 \times 160$ pixel images with a 128-colour palette
2. Pixel-wise maximum – flickering
3. Extract Y channel (luminance), rescale $84 \times 84$ $(\phi)$ – computation and memory
4. Frame skipping – Not much change, computation
5. Stack 4 frames to produce the input to the Q-function ($84 \times 84 \times 4$) – Sequence



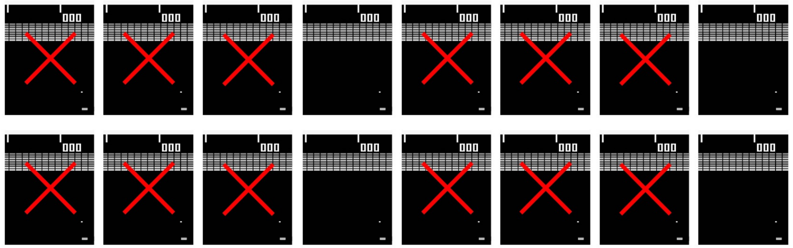
$210 \times 160$



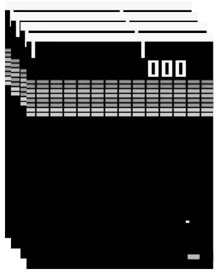
Pixel-wise maximum



Luminance, Rescale



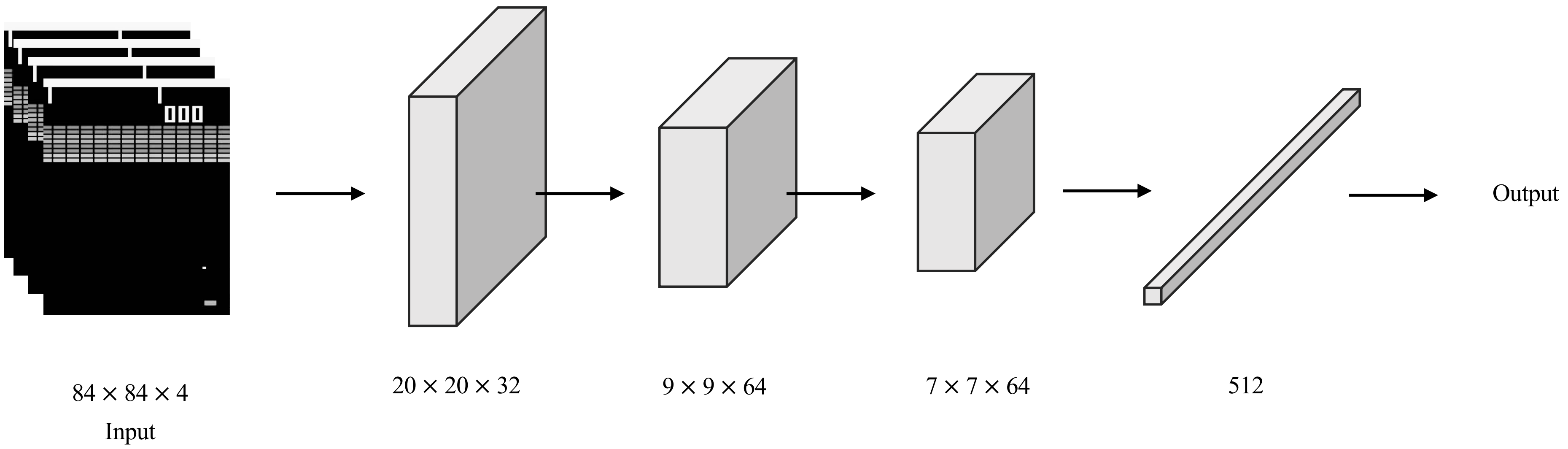
Frame skipping



Stack

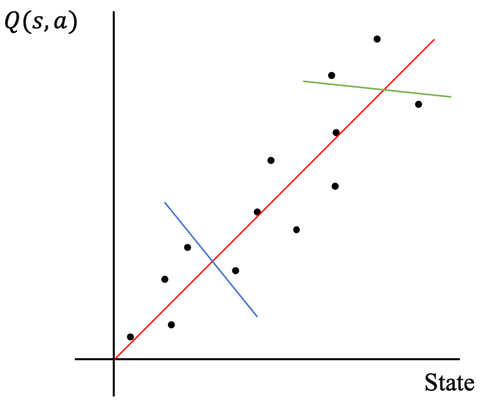
# DQN Model Architecture

- Input : Preprocessing ($\phi$)
- $1^{st}$ hidden layer : 32 filters, $8 \times 8$ with stride 4, ReLU
- $2^{nd}$ hidden layer : 64 filters, $4 \times 4$ with stride 2, ReLU
- $3^{rd}$ hidden layer : 64 filters, $3 \times 3$ with stride 1, ReLU
- Final hidden layer : FC layer 512 units, ReLU
- Output layer : Number of valid action

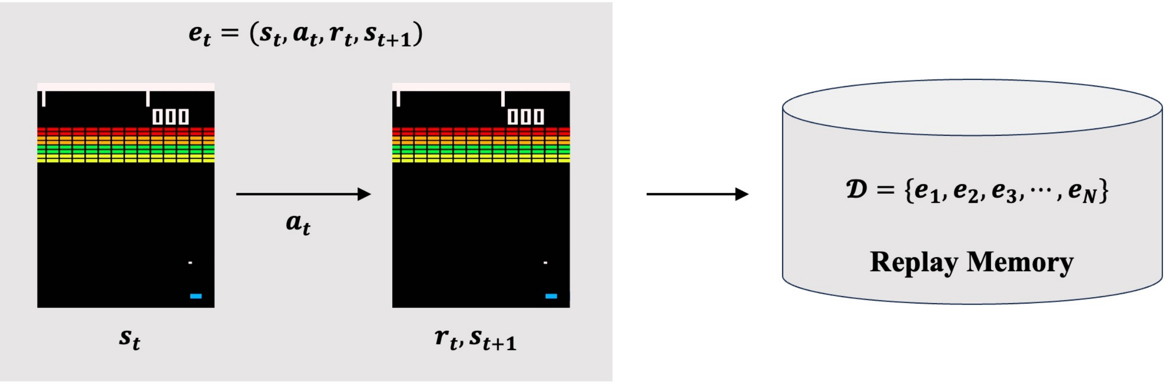


# Experience Replay

- Deep learning algorithms assume the data samples to be independent
- State sequences in RL are highly correlated
- Strong correlations between the samples
- Randomizing the samples breaks these correlations



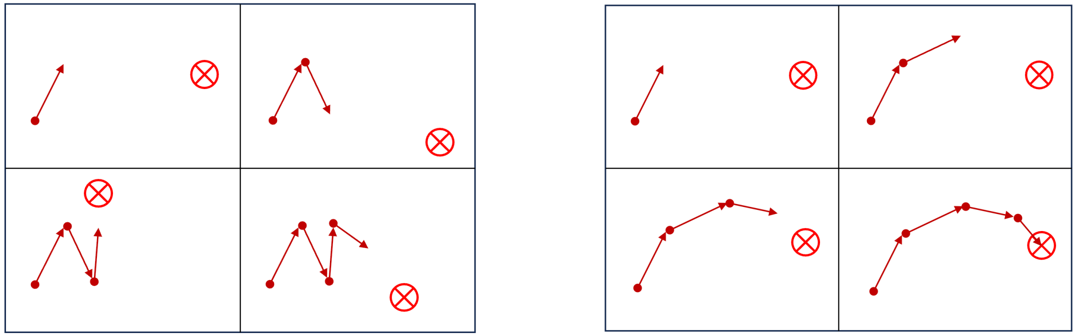

Correlation Problem




Experience Replay

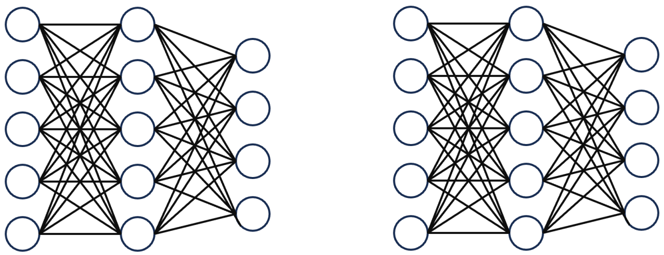
## Target Network

- Use a separate network for generating the targets $y_j$ in the Q-learning update
- Every C updates clone the network Q to obtain a target network $\hat{Q}$
- More stable compared to standard Q-learning
- Oscillations, hard to convergence



No Target Network

Target Network



$Q(s, a; \theta) \longrightarrow Q(s, a; \theta^{-})$

Main Network

Target Network

# DQN Pesudo-code (2015 NATURE)

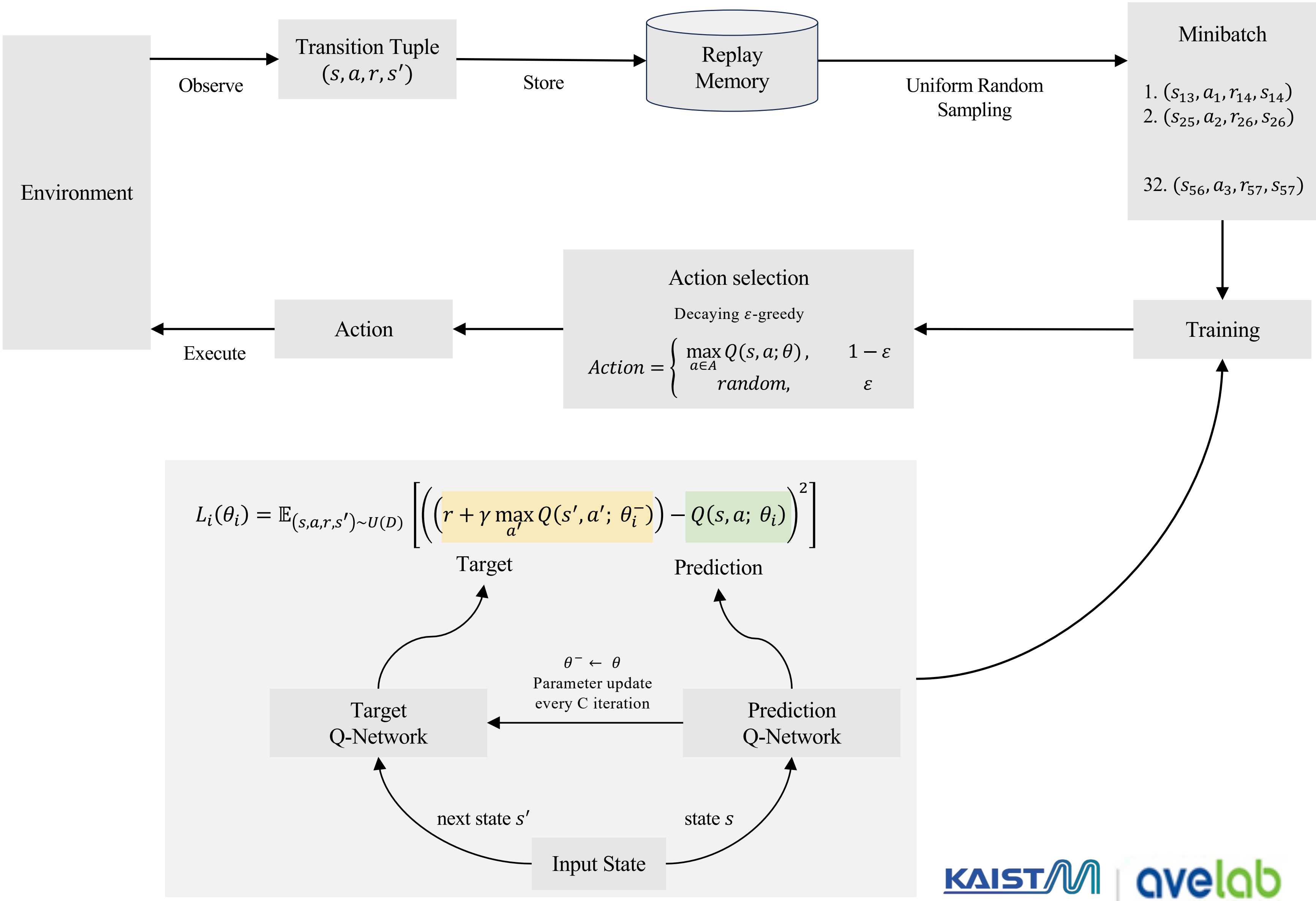
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
**For** episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$ Preprocessing
  **For** t = 1, T **do**
    With probability ε select a random action $a_t$
    otherwise select $a_t = \underset{a}{argmax}\, Q(\phi(s_t), a; \theta)$ ε-greedy
    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 $\mathcal{D}$ Experience Replay

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

Target Network

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters $\theta$
    Every C steps reset $\hat{Q} = Q$
  **End for**
**End for**

# DQN Diagram



Environment
Observe
Transition Tuple
$(s, a, r, s')$
Store
Replay Memory
Uniform Random Sampling
Minibatch
1. $(s_{13}, a_1, r_{14}, s_{14})$
2. $(s_{25}, a_2, r_{26}, s_{26})$
32. $(s_{56}, a_3, r_{57}, s_{57})$
Training
Action selection
Decaying $\varepsilon$-greedy
$Action = \begin{cases} \max_{a \in A} Q(s, a; \theta), & 1-\varepsilon \\ random, & \varepsilon \end{cases}$
Action
Execute
$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[ \left( \left( r + \gamma \max_{a'} Q(s', a'; \theta_i^-) \right) - Q(s, a; \theta_i) \right)^2 \right]$
Target
Prediction
$\theta^- \leftarrow \theta$
Parameter update every C iteration
Target Q-Network
Prediction Q-Network
next state $s'$
state $s$
Input State

# DQN Hyperparameter

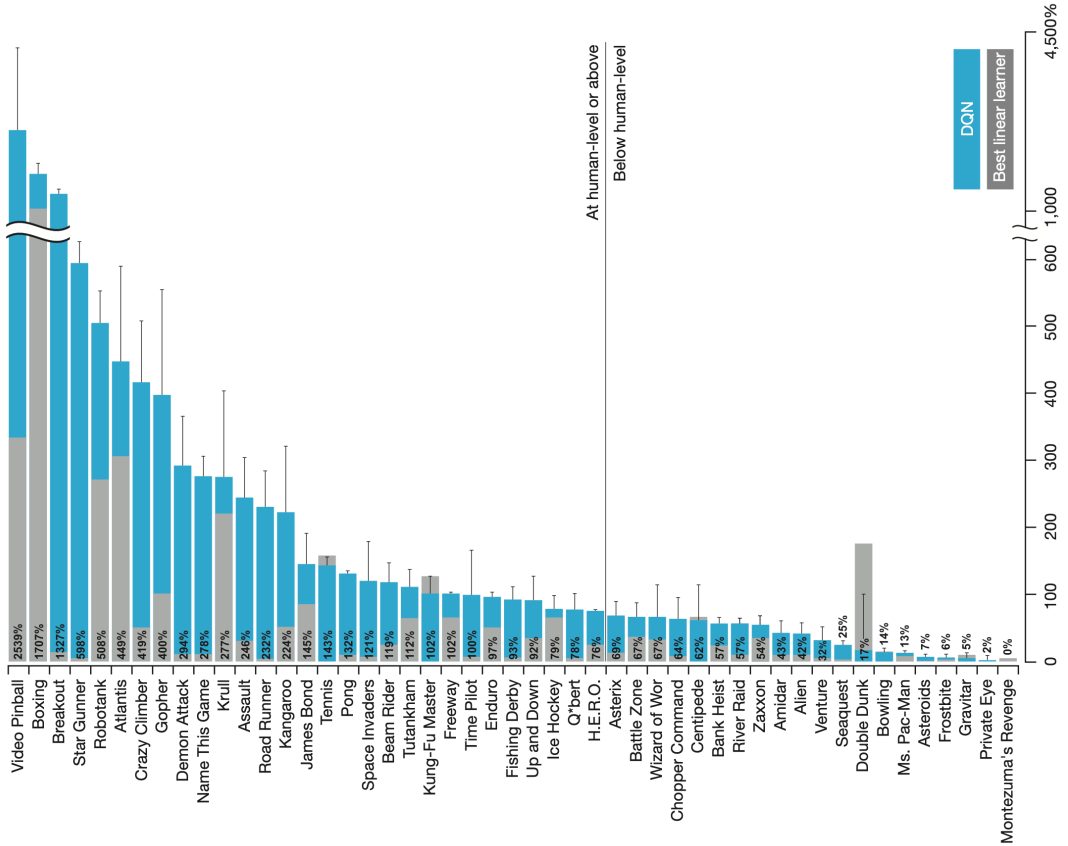
| Hyperparameter | Value | Description |
|---|---|---|
| minibatch size | 32 | Number of training cases over which each stochastic gradient descent (SGD) update is computed. |
| replay memory size | 1000000 | SGD updates are sampled from this number of most recent frames. |
| agent history length | 4 | The number of most recent frames experienced by the agent that are given as input to the Q network. |
| target network update frequency | 10000 | The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter *C* from Algorithm 1). |
| discount factor | 0.99 | Discount factor gamma used in the Q-learning update. |
| action repeat | 4 | Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame. |
| update frequency | 4 | The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates. |
| learning rate | 0.00025 | The learning rate used by RMSProp. |
| gradient momentum | 0.95 | Gradient momentum used by RMSProp. |
| squared gradient momentum | 0.95 | Squared gradient (denominator) momentum used by RMSProp. |
| min squared gradient | 0.01 | Constant added to the squared gradient in the denominator of the RMSProp update. |
| initial exploration | 1 | Initial value of $\varepsilon$ in $\varepsilon$-greedy exploration. |
| final exploration | 0.1 | Final value of $\varepsilon$ in $\varepsilon$-greedy exploration. |
| final exploration frame | 1000000 | The number of frames over which the initial value of $\varepsilon$ is linearly annealed to its final value. |
| replay start size | 50000 | A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory. |
| no-op max | 30 | Maximum number of "do nothing" actions to be performed by the agent at the start of an episode. |

| Hyperparameter | Value |
|---|---|
| minibatch size | 32 |
| Replay memory size | 1,000,000 |
| Target network update frequency | 10,000 |
| Initial exploration | 1 |
| Final exploration | 0.1 |
| Replay start size | 50,000 |

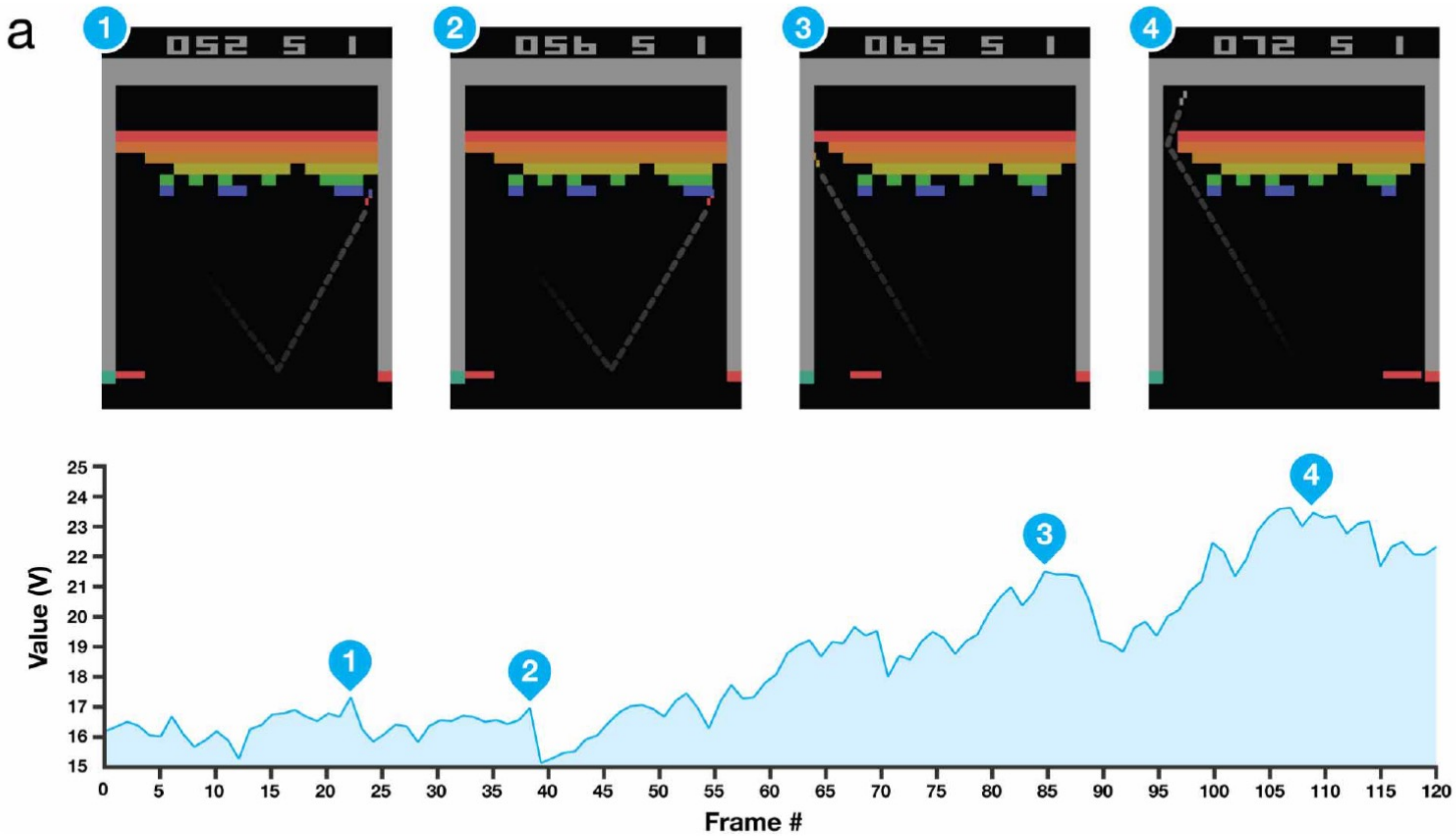
List of hyperparameters and their values

# Result – Atari 2600

- Same network architecture, learning algorithm and hyperparameter setting
- Each game 30 times for up to 5min
- 100% were professional human game testers, 0% were random play
- Outperformed humans in 29 out of 49



# Result – Breakout



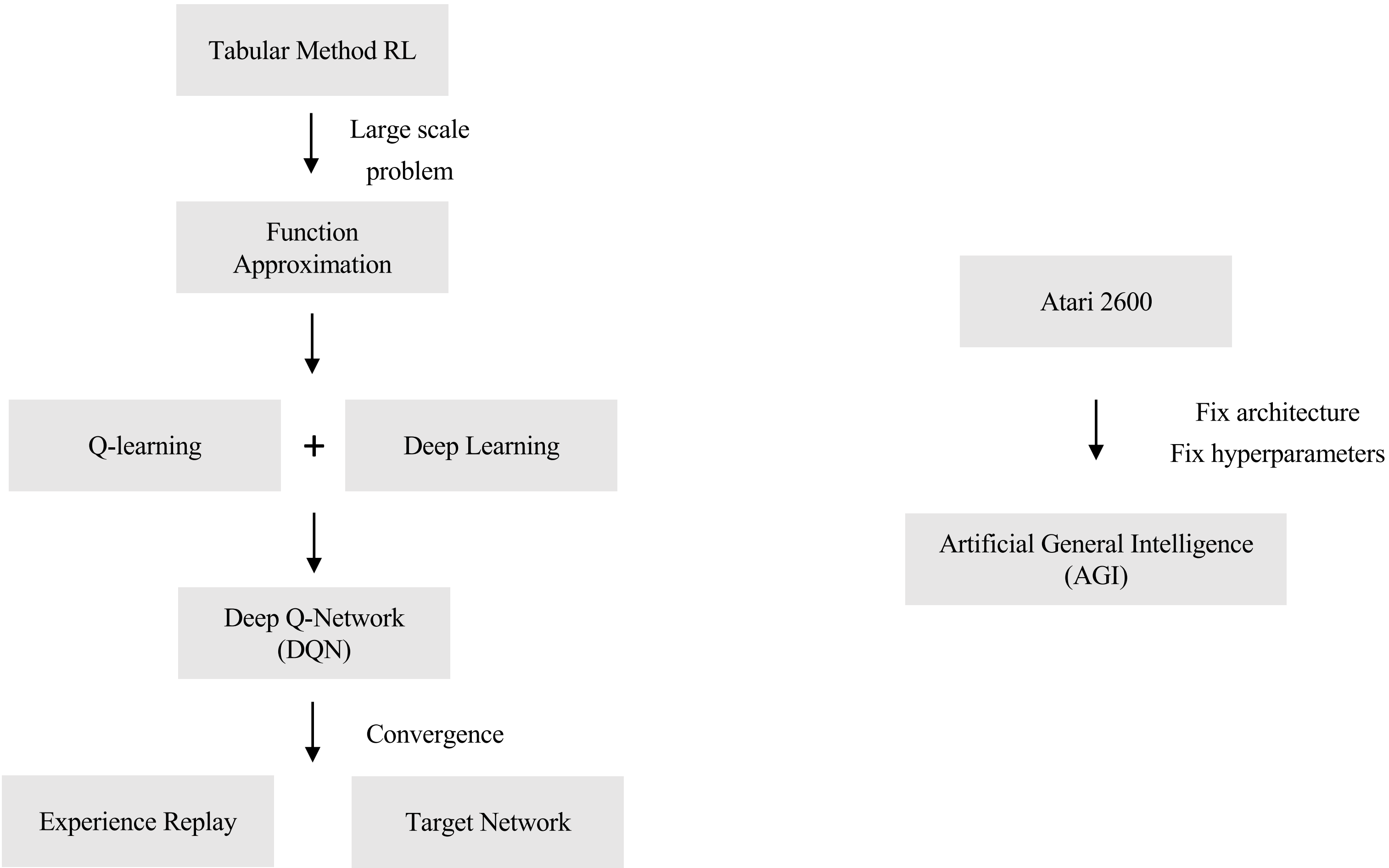
# Result – Effect of ‘Experience Replay’ and ‘Target Network’

| 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 |

## Result – Effect of Neural Network

| Game | DQN | Linear |
|---|---|---|
| Breakout | 316.8 | 3.00 |
| Enduro | 1006.3 | 62.0 |
| River Raid | 7446.6 | 2346.9 |
| Seaquest | 2894.4 | 656.9 |
| Space Invaders | 1088.9 | 301.3 |

# Overview


Tabular Method RL
Large scale
problem
Function
Approximation
Q-learning
+
Deep Learning
Deep Q-Network
(DQN)
Convergence
Experience Replay
Target Network
Atari 2600
Fix architecture
Fix hyperparameters
Artificial General Intelligence
(AGI)