用領域知識提升DQN學習效率之研究

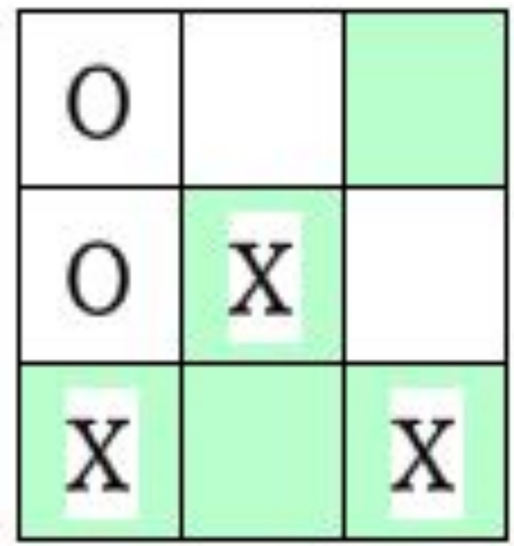
Use Domain Knowledge to Make DQN Learn Faster for Tic-Tac-Toe Game

何佩蓁
台北市私立東山高中

楊宗憲 博士
賽微科技股份有限公司

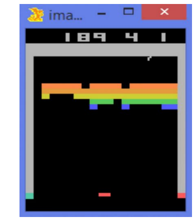
Objectives

- Use **Deep Q Networks** (DQN) to achieve unbeatable Tic-Tac-Toe.
- Use **domain knowledges** to make DQN utilize training data more efficiently.



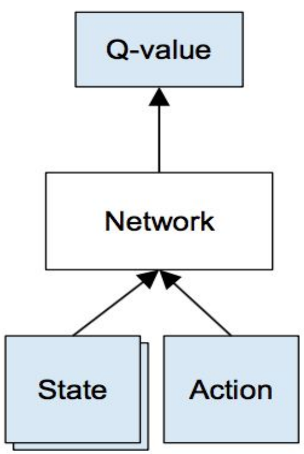
Deep Q Networks

- **DQN = Q-Learning + NN (DeepMind 2013, 2015)**
  - $Q(s,a)$ approximated by Neural Networks
  - $Q(s) \rightarrow$ a list of Q-values for all actions, for speedup
  - Experience Replay: random sampling from recently collected training data.

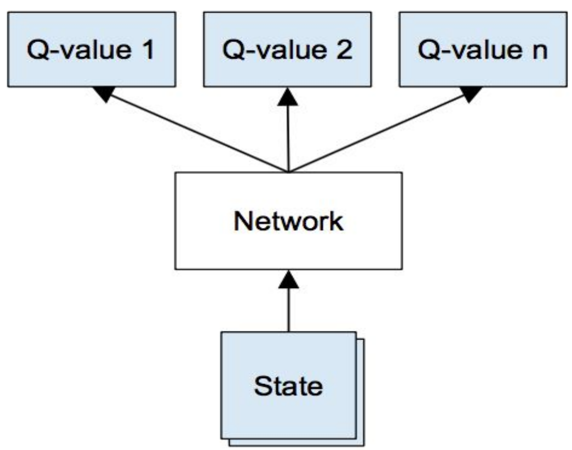


Atari 2600 games

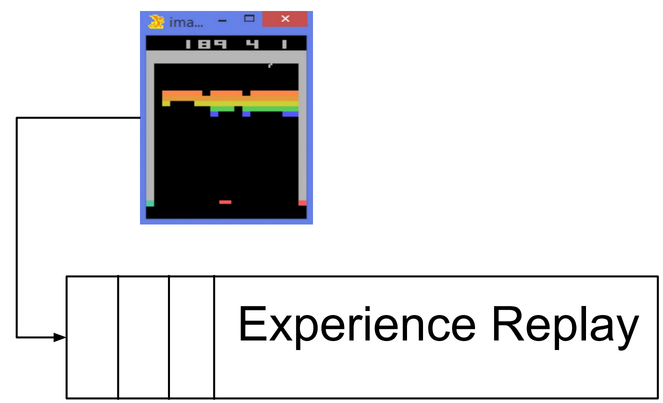
http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/



Original Q function



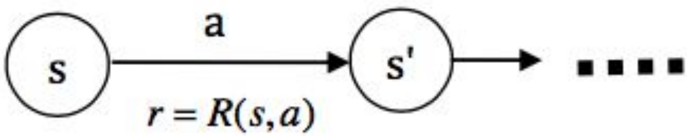
DeepMind's NN approximated Q function



- **Bellman Estimates target *Q*^{t} from (*s*, *a*, *r*, *s*')**

$$Q(s;\theta) \rightarrow Q = [q_1, q_2, \ldots, q_n]$$

$$Q(s';\theta^-) \rightarrow Q' = [q'_1, q'_2, \ldots, q'_n]$$



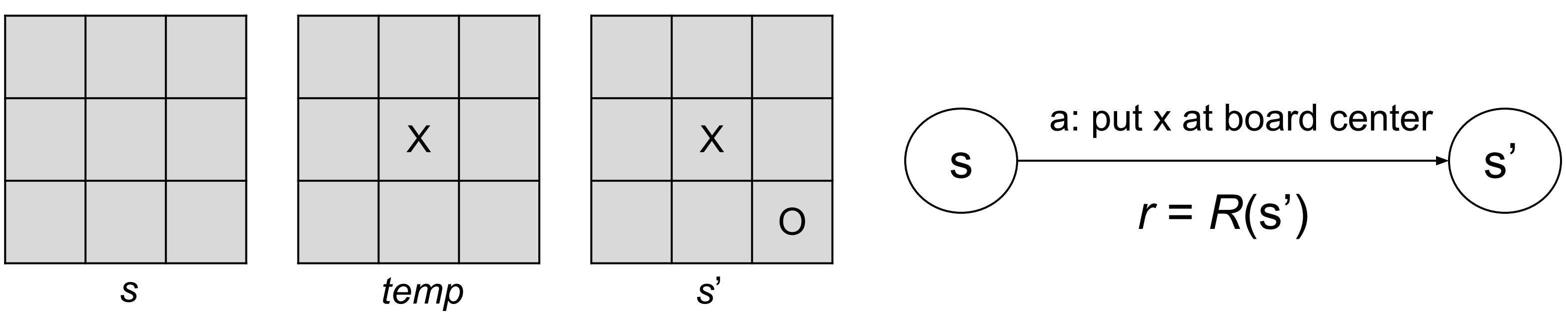
$$q_a^t = \begin{cases} r, & \text{if } s' \in S_T \\ r + \gamma \times \max_i q'_i, & \text{else} \end{cases}$$

$$Q^t \leftarrow Q$$

$$Q^t[a] \leftarrow q_a^t \quad \Rightarrow \quad Q^t = [q_1, q_2, \ldots \boxed{q_a^t}, \ldots q_n]$$

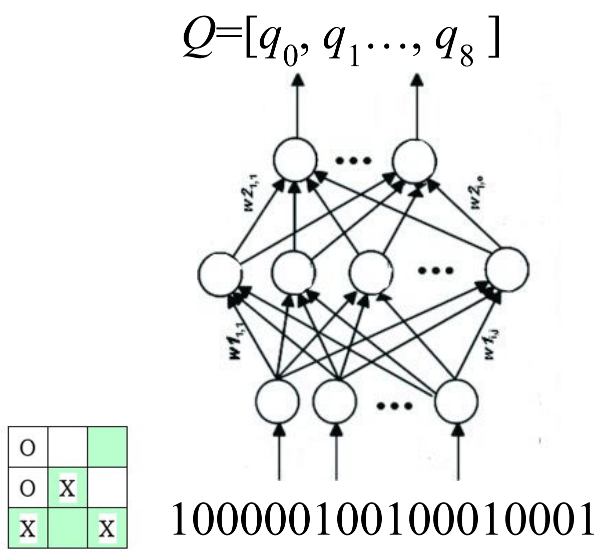
DQN Implementing Tic-Tac-Toe

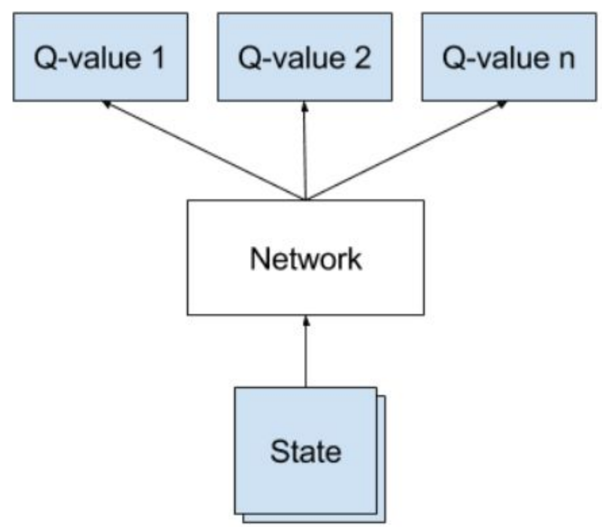
- **Markov Decision Process**
  - Stochastic environment: opponent may move randomly
  - *s* : current board state
  - *a* : 9 board locations to put pieces
  - $$r = R(s') = \begin{cases} 1 & \text{if DQN wins} \\ -1 & \text{if DQN loses} \\ 0.1 & \text{tie} \\ 0 & \text{game not finished} \end{cases}$$
  - *s'* : the board state after opponent puts the piece



MLP *Q* Function

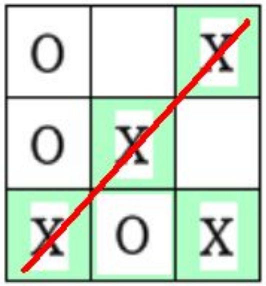
- Input: 2 bits for each square on board,totally 9*2=18 neurons.
- Hidden: one layer, 36 neurons, tanh(x)
- Output: 9 neurons, f(x)=x, for *Q* values of 9 squares on board.



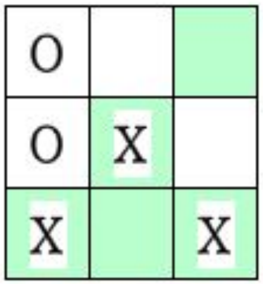


Make DQN Learn Faster

- **Extra copies of determinative transitions into Experience Replay**
  - For (s, a, r, s'), s' is terminal state.
  - Higher probability for determinative transitions to be selected as training samples



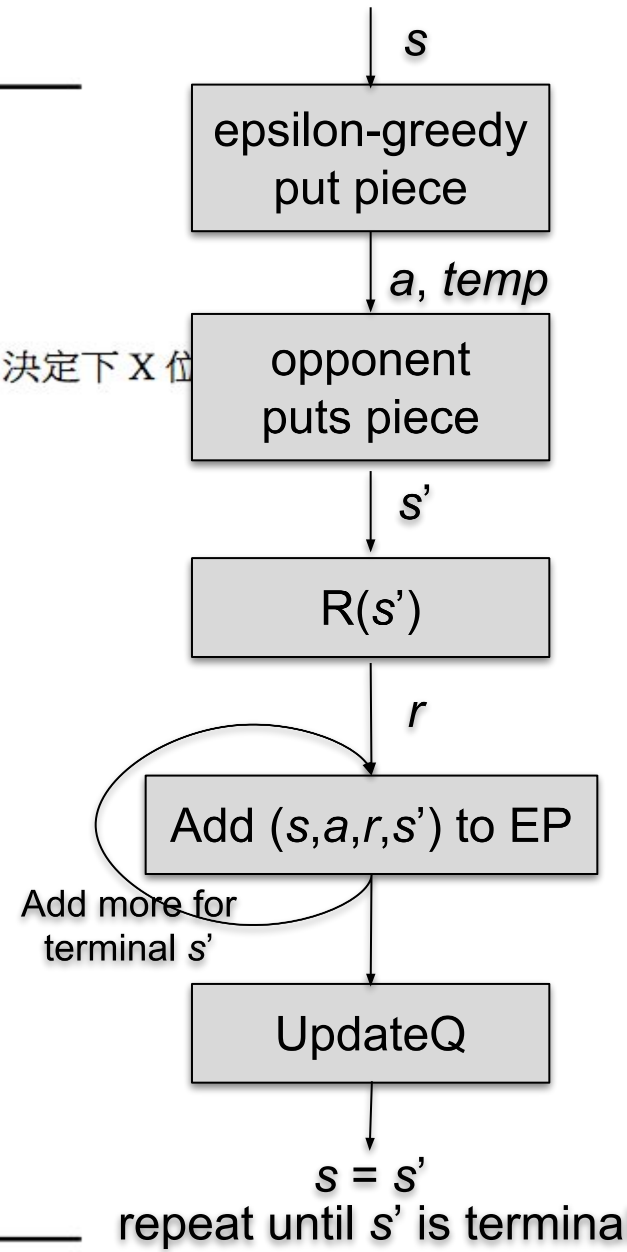
- **Early rewarding for 2-way winning board**
  - DQN learns to deliberately move toward 2-way winning board states to win the game.



演算法 1：用 DQN 訓練井字遊戲先手下 X

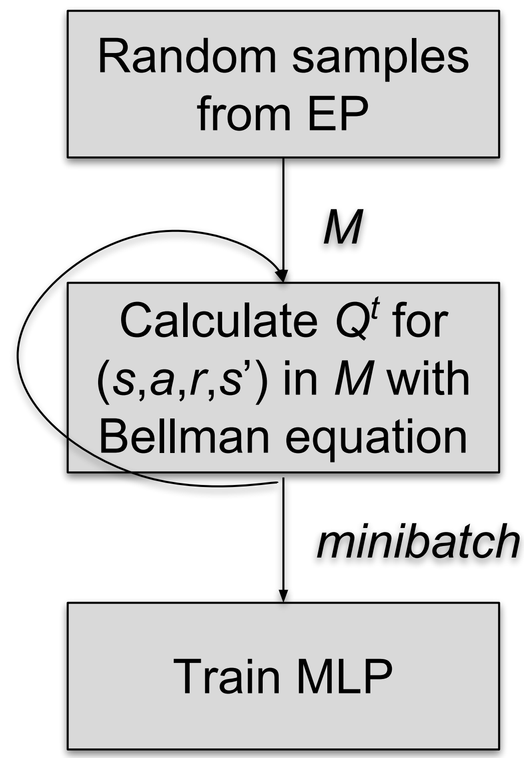
```
以隨機值初始化 MLP 參數 θ
θ⁻ ← 複製 θ
for episode = 1, 2, …, T do
    s ← 初始化空盤面
    while s 非終局盤面 do
        a ← 以 ε − greedy 方式隨機或 arg max MLP(s; θ) 決定下 X 位
        temp ← 先手於盤面 s 的 a 位置下 X
        if temp 為終局盤面 then
            s' ← temp
        else
            s' ← 對手在盤面 temp 隨機下 O
        end if
        r = R(s')
        將 (s, a, r, s') 加入經驗回放
        if s' 為終局盤面 then
            增加 (s, a, r, s') 加入經驗回放的數量
        end if
        θ ← UpdateQ(θ, θ⁻)
        s ← s'
    end while
    每隔若干次 episode 更新 θ⁻ ← 複製 θ
    每隔若干次 episode 衰減 ε ← 0.9 × ε
end for
```

move one piece



演算法 2：更新 MLP 的參數 θ

function $UpdateQ(\theta,\theta^{-})$
    從經驗回放隨機挑選若干筆資料放入 M ，包含最新加入的該筆資料
    $minibatch \leftarrow$ 空串列
    for each (s,a,r,s') in M do
        $MLP(s;\theta) \rightarrow Q=[q_0,q_1,\ldots,q_8]$
        if s' 為終局盤面 then
            $q_a^t = r$
        else
            $MLP(s';\theta^{-}) \rightarrow Q'=[q'_0,q'_1,\ldots,q'_8]$
            $q_a^t = r+\gamma \max_{a'}(q'_{a'}, s'[a'] \notin \{O,X\})$
        end if
        $Q^t \leftarrow Q$
        $Q^t[a]=q_a^t$
        將 (s,Q^t) 加入 $minibatch$
    end for
    $\theta \leftarrow TrainMLP(\theta,minibatch)$
    return θ



Experiments

- **Experimental environment**
  - Training episodes: 20k for 1st mover, 100k for 2nd
  - MLP Learning rate α : $\alpha = 0.01$, $\alpha \leftarrow \alpha \times 0.9$, decay 50 times
  - ε-greedy : $\varepsilon = 1$, $\varepsilon \leftarrow \varepsilon \times 0.9$, decay 50 times
  - Experience Replay capacity : 2048 , minibatch size: 32
  - 10 identical experiments for each setting, randomly initialize θ
  - Store θ for every 100 training episodes
- **Test Criterion**
  - Unbeatable, playing Tic Tac Toe against randomly moving program for 10,000 rounds without losing any game.

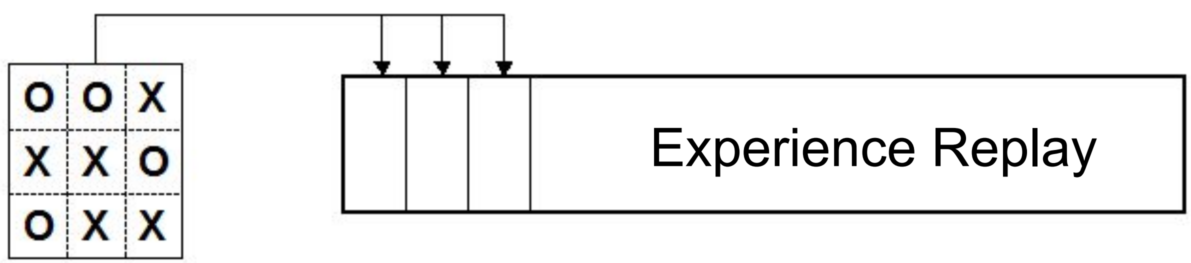
Experiment 1

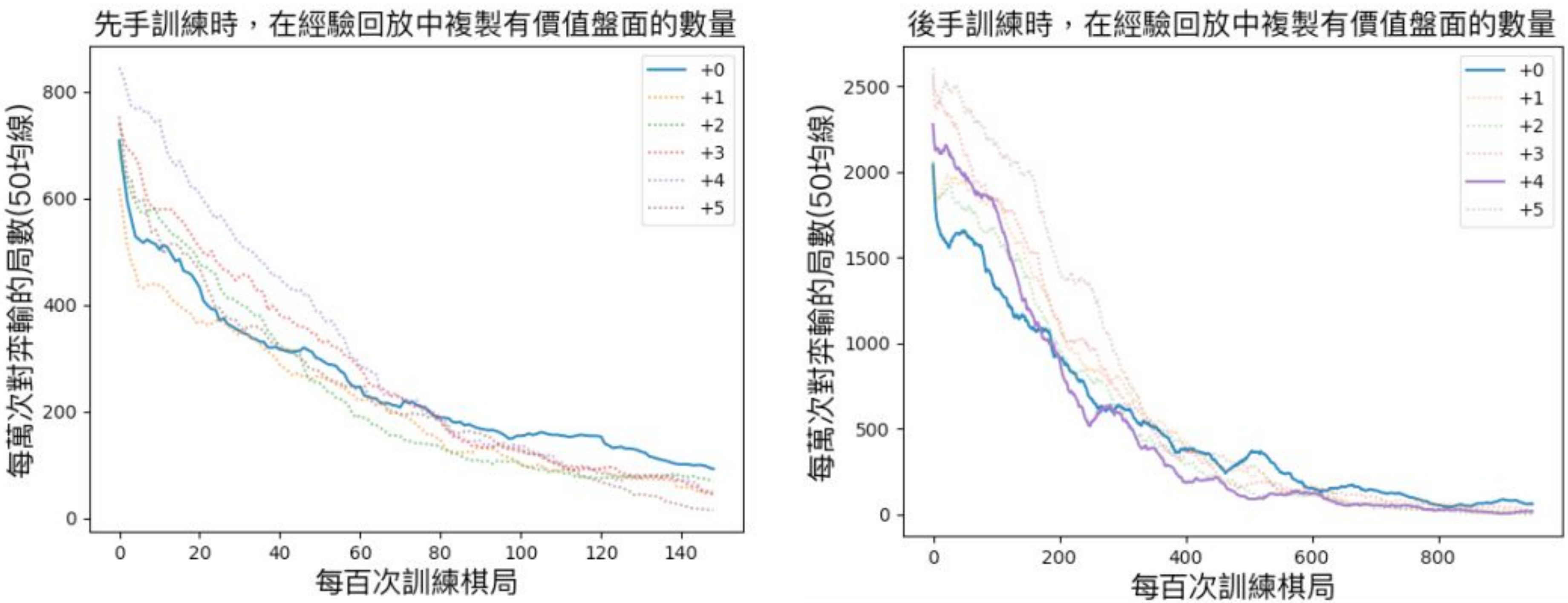
- **Adding extra copies of determinative transitions to Experience Replay**

Average number of episodes for 1st and 2nd movers to reach unbeatable

extra copies	0	1	2	3	4	5
1st mover	9.7k	8.9k	11.0k	10.5k	11.3k	9.6k
2nd mover	$\boxed{82.5k}$	68.9k	72.0k	62.0k	$\boxed{58.0k}$	65.6k

30% reduction





For both DQN trained player 1 and 2, adding extra copies of determinative transitions to Experience Replay can make learning faster during the training progress, as compared with the baselines.

Experiment 2

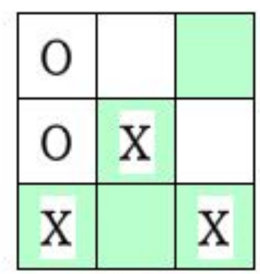
- **Early rewarding for 2-way winning for 1st mover**

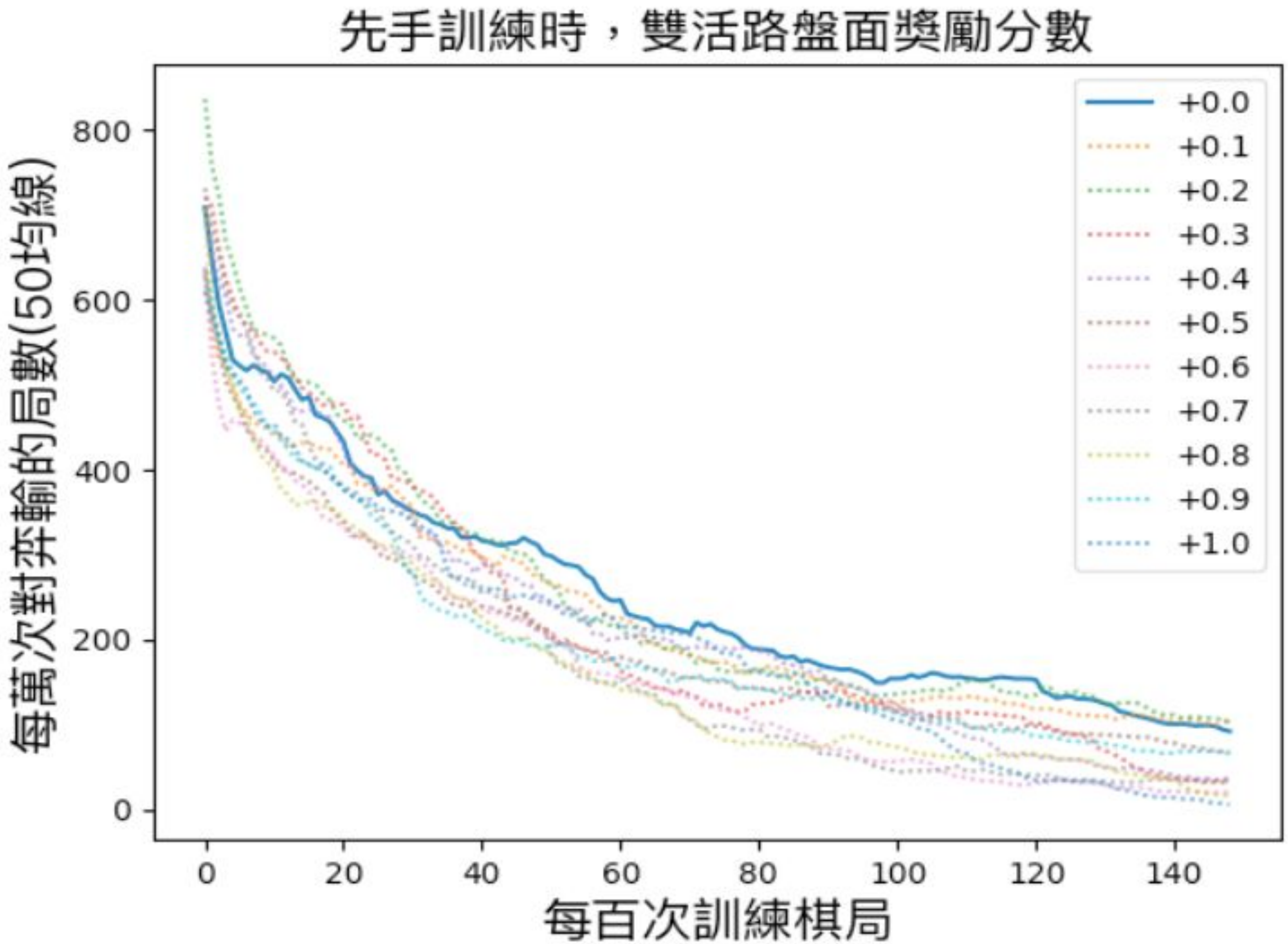
Numbers of episodes for 1st mover to reach unbeatable after adding reward

Reward	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Episodes	9.7k	10.2k	8.8k	9.4k	9.2k	8.9k	7.9k	7.8k	8.5k	7.0k	8.5k

28% reduction

  - When the reward is higher than 0.5, it starts to improve significantly in terms of reaching unbeatable criterion.





In terms of learning speed, offering extra rewards for two-way winning board outperforms the baseline noticeably no matter how much the reward is.

Experiment 3

- **Cutting training data in half when training 2nd mover**
  - Reducing to 50k training episodes
  - Due to the second mover's disadvantage, DQN training may fail to reach unbeatable criterion without sufficient training episodes.
  - Result: improving training reliability by nearly 100%.

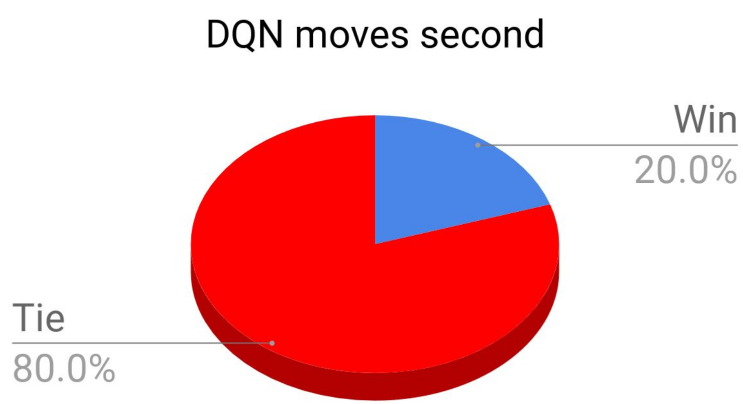
The effect on learning stability of DQN when adding extra copies

extra copies	0	1	2	3	4	5
reach unbeatable (%)	40	80	70	70	80	80

improve nearly 100%

Experiment 4

- **Playing against human players**
  - To ensure that DQN trained players can really reach the goal of being unbeatable (without losing any game).
  - 10 testers of ages above 15, each plays 2 rounds.
  - DQN moves first: two-way winning reward **0.9**
  - DQN moves second: **4** extra copies of determinative transitions



Experiment 5: DQN Self Playing

	X	

O		
	X	

O		
	X	
X		

O		O
	X	
X		

O	X	O
	X	
X		

O	X	O
	X	
X	O	

O	X	O
	X	
X	O	X

O	X	O
	X	O
X	O	X

O	X	O
X	X	O
X	O	X

X: two-way winning reward **0.9**

O: **4** extra copies of determinative transition

Conclusion

- **Summary**
  - A DQN approach using MLP approximated Q function is proposed to train a computer to play Tic-Tac-Toe invincibly.
  - Domain knowledges can make DQN utilize training data more efficiently, including
    - Adding extra copies of determinative training samples into Experience Replay
    - Early rewarding for some specific game states that can lead to winning
- **More Domain Knowledges**
  - While training 2nd mover, give penalty when the 1st mover gets to the 2-way winning board.