第十九屆旺宏科學獎

創意說明書

參賽編號：SA19-191

作品名稱：Algorithms to play UNO

姓名：吳邦寧

關鍵字：基因算法、強化學習、機器學習

摘要

電腦攻克了許多領域，包括西洋棋、圍棋以及五子棋，卻鮮少有人研究UNO──全世界最暢銷的桌遊。

因此，本研究設計了五種不同的演算法，Dummy algorithm、Naïve offensive algorithm、Naïve defensive algorithm、Genetic algorithm以及Deep-Q-Learning algorithm，並讓這五種演算法玩UNO，相互對抗，看誰是最終贏家。

經過實驗得知，UNO是一個「先手優勢」的遊戲，不論是哪種傳統演算法，先手的勝率都比較高，為了保證實驗結果正確性，其餘關於勝率的實驗都是做先後手各三十場，確保數據不受先後手影響。

每種演算法都有自己的特性，Dummy algorithm是一個名副其實的「假人」，不論怎麼打都必輸，正如預料；Naïve offensive algorithm如同只見近利之人，有好牌就出，被Naïve defensive algorithm，也就是「永遠最後才出好牌」的人，給打敗了。

Genetic algorithm好比細菌，使用基因記憶所學所能；Deep-Q-Learning好比人類，使用大腦記憶所學所能，若是細菌對上人類，也就是Genetic algorithm對上Deep-Q-Learning，究竟會如何呢？答案是人類獲勝，就像人類在瘟疫中戰勝細菌一樣，Deep-Q-Learning也戰勝了Genetic algorithm，不過偶爾也有Genetic algorithm戰勝Deep-Q-Learning的時候，就像是瘟疫爆發，細菌偶爾也有戰勝人類的時候。

本研究設計了諸多演算法，經過實驗後得知，這些參賽者中的黑馬是Deep-Q-Learning，不僅勝率穩定，還能夠藉由Behavior cloning增加勝率，是目前最強的演算法。

繼Deep Blue、Alpha Go之後，電腦又將矛頭指向全世界最暢銷的桌遊──UNO，蓄勢待發，準備攻克這個領域！

壹、研究動機

電腦攻克了許多領域，包括西洋棋、圍棋以及五子棋，卻鮮少有人研究UNO──全世界最暢銷的桌遊。

經過網路文獻回顧，只有一篇「Notes on machine learning - Playing UNO」提出了相關的研究，該文作者採用Genetic algorithm進行實驗，然而成效不彰，且作者無意繼續研究。

細菌好比Genetic algorithm，人類好比Deep-Q-Learning；細菌只能藉由基因記憶所學所能，但是人類能夠藉由大腦記憶所學所能，若是使用Deep-Q-Learning進行研究，必能帶來斬獲。

Deep-Q-Learning常常應用在Unsupervised learning & Reinforcement learning領域，Alpha Go也是基於Deep-Q-Learning的產物，筆者認為若是Deep-Q-Learning與Genetic algorithm較勁，Deep-Q-Learning將會勝出，就好比人類在一次又一次的瘟疫中戰勝細菌。

貳、研究目的

一、設計各種不同的演算法來玩UNO

二、探討先手與後手的利與弊

三、探討各演算法之間的勝率

四、探討Deep-Q-Network與Genetic-Network的利與弊

參、文獻回顧

# 一、Notes on machine learning - Playing UNO

<https://john-hearn.info/articles/notes-on-machine-learning-playing-uno>

The article used four different algorithms, dummy algorithm, defensive algorithm, offensive algorithm and genetic algorithm to conduct the research. The author also reveals that genetic optimization is not effective on deep neural networks due to the randomness of genetic fluctuation.

# 二、RL - Cards

<https://arxiv.org/pdf/1910.04376.pdf>

It is an open source library recently published in Feb 2020 for conducting reinforcement learning researches of card games. Despite its usefulness, the library could not meet my needs. Thus, I decided to use RL Cards as a blueprint to build on my own.

# 三、Deep-Q-Learning

<https://www.mlq.ai/deep-reinforcement-learning-q-learning/>

The above is an article with tutorial of Deep-Q-Learning illustrated by vivid graphs and simple math, which is much more unsophisticated than the original paper. The article would guide the reader to understand the concepts of neural networks, Deep-Q-Learning and Markov decision process.

肆、研究器材

|  |  |  |
| --- | --- | --- |
|  | 器材 | 用途 |
| 研究器材 | Tensorflow | Decrease the coding complexity for developing. |
| Tkinter | Visualize the gaming process. |
| NVIDIA 1660 Ti | Hardware for increasing the training performance. |
| CUDA | Utilize GPU to increase training performance. |
| Tensorboard | Plot the win rate. |
| Python | Major programming language. |
| Github | Online web service for storaging codes and files. |

伍、研究方法

# 一、遊戲規則

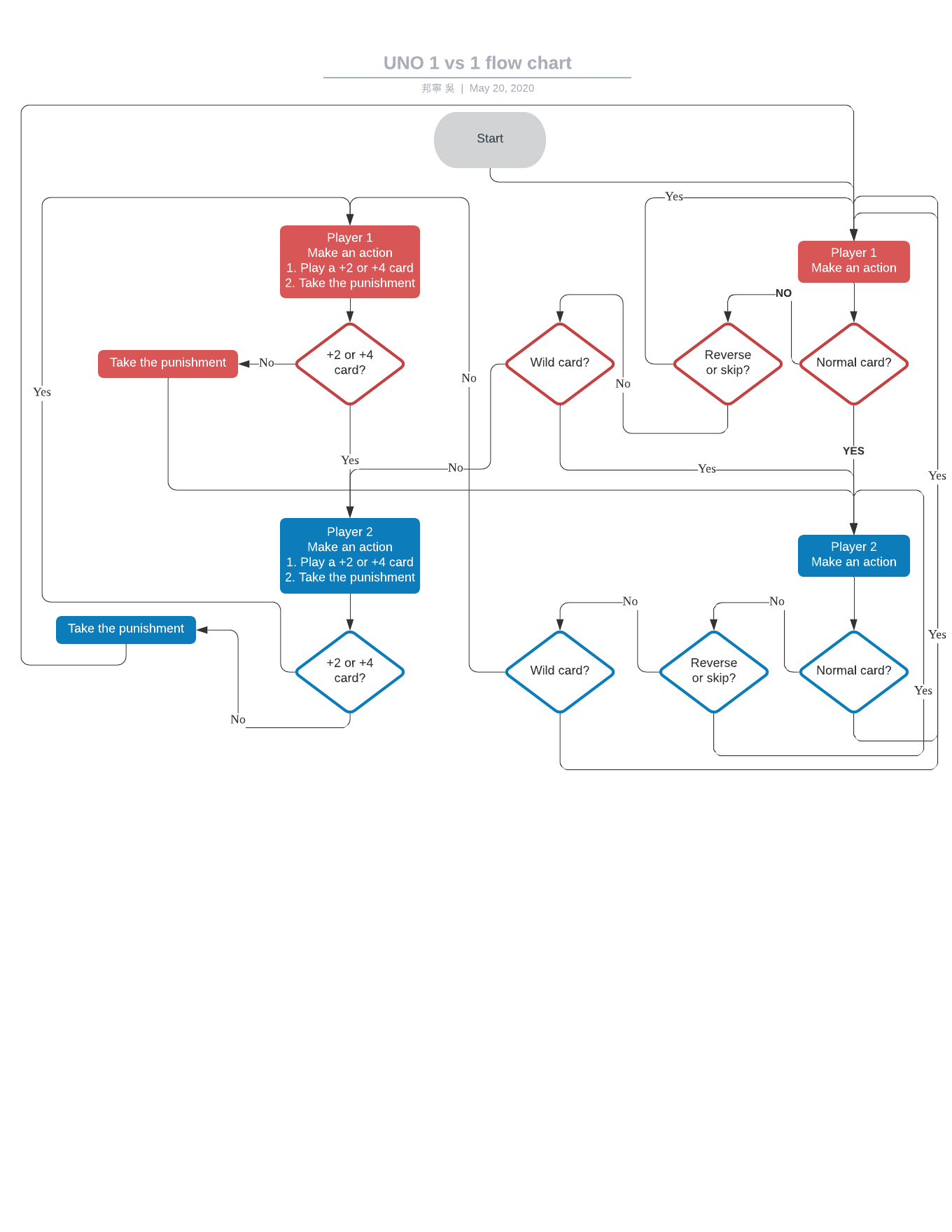
一張圖勝過千言萬語，如下簡易流程圖所示。

圖 1 ▲ 遊戲簡易流程圖

這並不是一張完整的流程圖，這張流程圖只是很簡明扼要的表達遊戲流程，像是「先出完牌就贏了」這種眾所皆知的規則沒有被記錄上去，值得注意的是，規則中不包含「Wild draw four challenge」。

# 二、實驗流程

## (一)、傳統演算法對抗傳統演算法

1. 讓兩個演算法進行遊戲，先後手至少各三十場

2. 統計結果，並回到 (1.)

## (二)、傳統演算法對抗神經網路

1. 初始化若干個神經網路，若是DQN則執行Behavior cloning

2. 讓神經網路與傳統演算法進行遊戲若干場

3. 使用勝場最多的神經網路與傳統演算法對抗先後手各三十場

4. 統計 (3.) 的結果，並回到 (2.)

## (三)、Deep-Q-Learning對抗Genetic algorithm

1. 初始化若干個Deep-Q-Network，並執行Behavior cloning

2. 讓DQN與假想敵(Defensive algorithm as default)進行遊戲若干場

3. 讓培養皿(Petri dish)上的Cells與勝場最多的DQN進行遊戲若干場

4. 使用勝場最多的Cell與勝場最多的DQN對抗先後手各三十場

5. 統計 (4.) 的結果，並回到 (3.)

## (四)、勝率圖表判讀方法

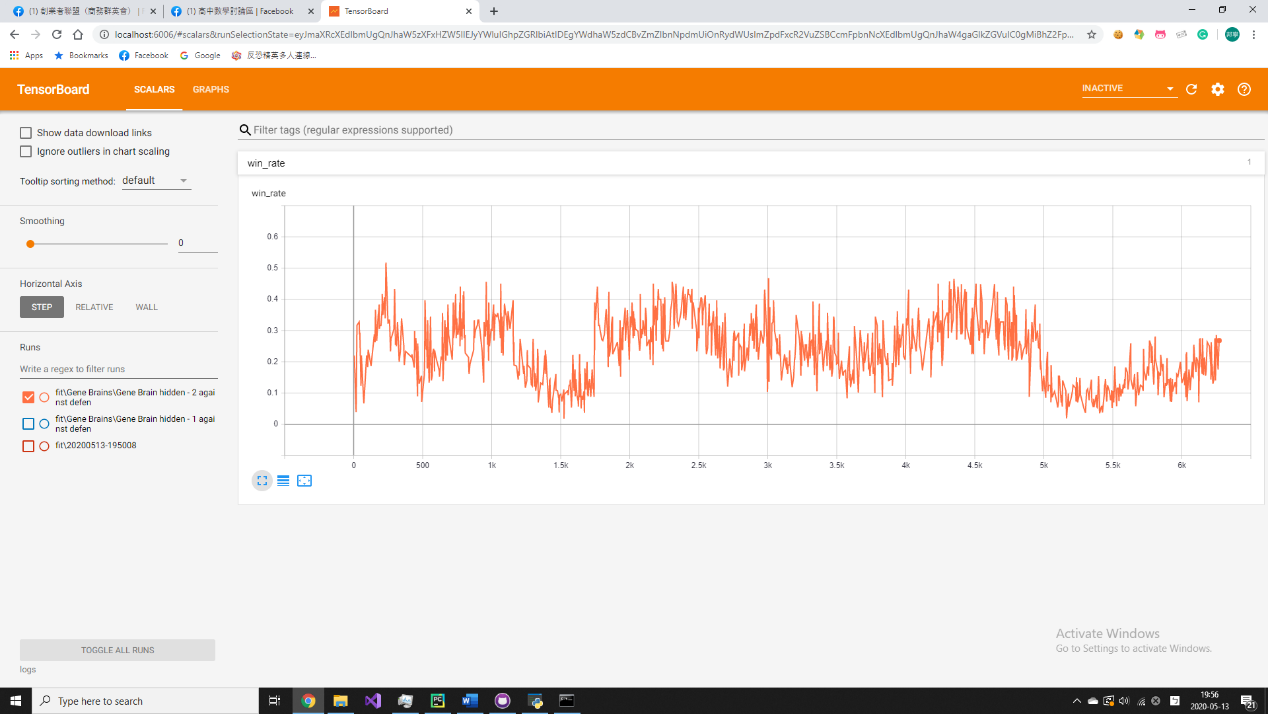
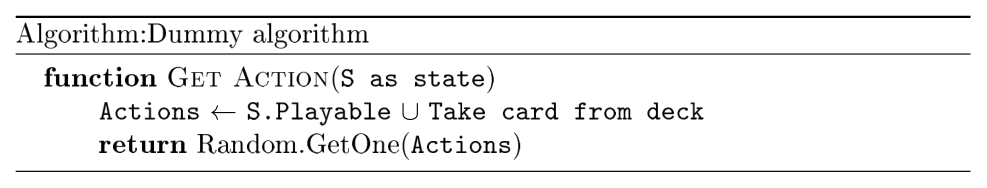
 下圖中，每一個點代表先後手各三十場，橫軸代表，縱軸代表勝率。

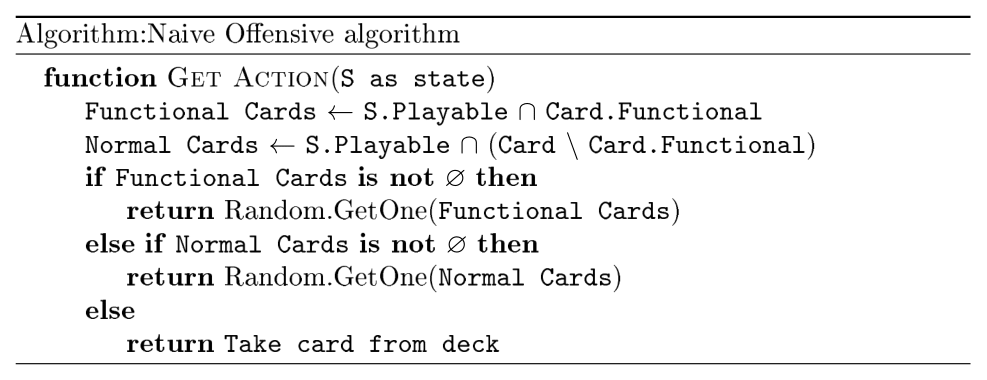
圖 2 ▲ 勝率圖範例

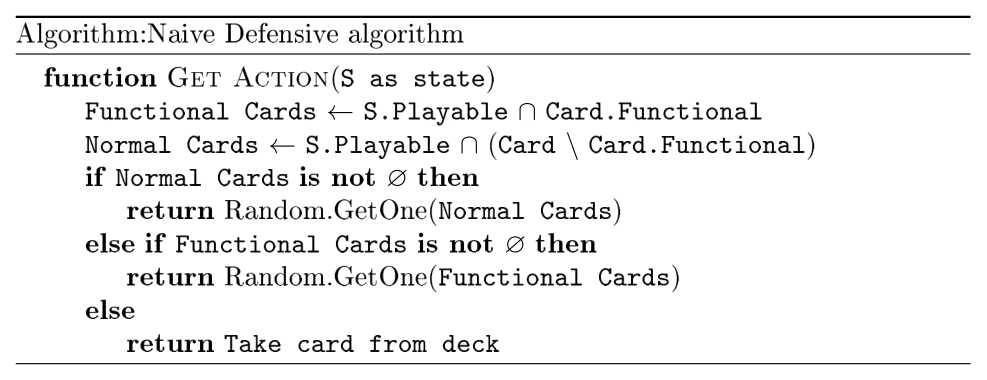
# 三、Dummy algorithm

一言以蔽之，這個演算法等同「找一張能丟的卡或抽牌」。

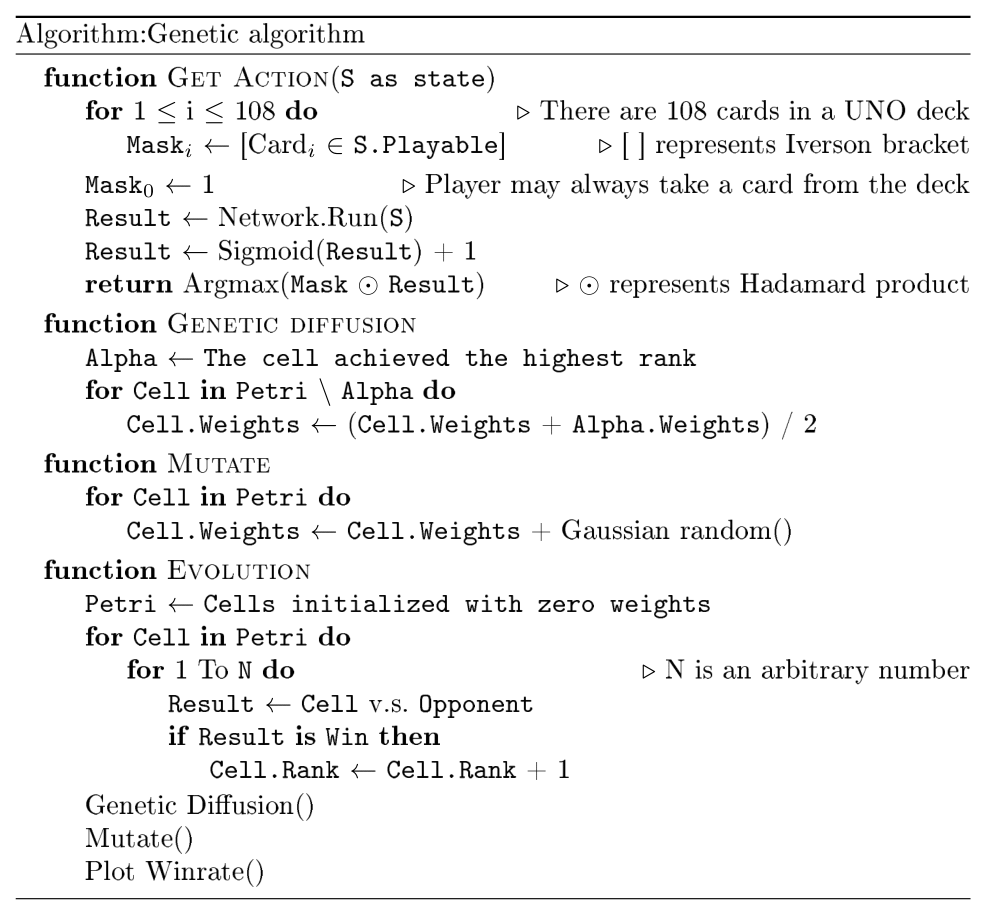


# 四、Naïve Offensive/Defensive Algorithm

Offensive algorithm等同「先丟最強的功能牌，再丟普通牌，沒牌再抽卡」，猶如忿狷之人，有好牌就出，不顧後果。

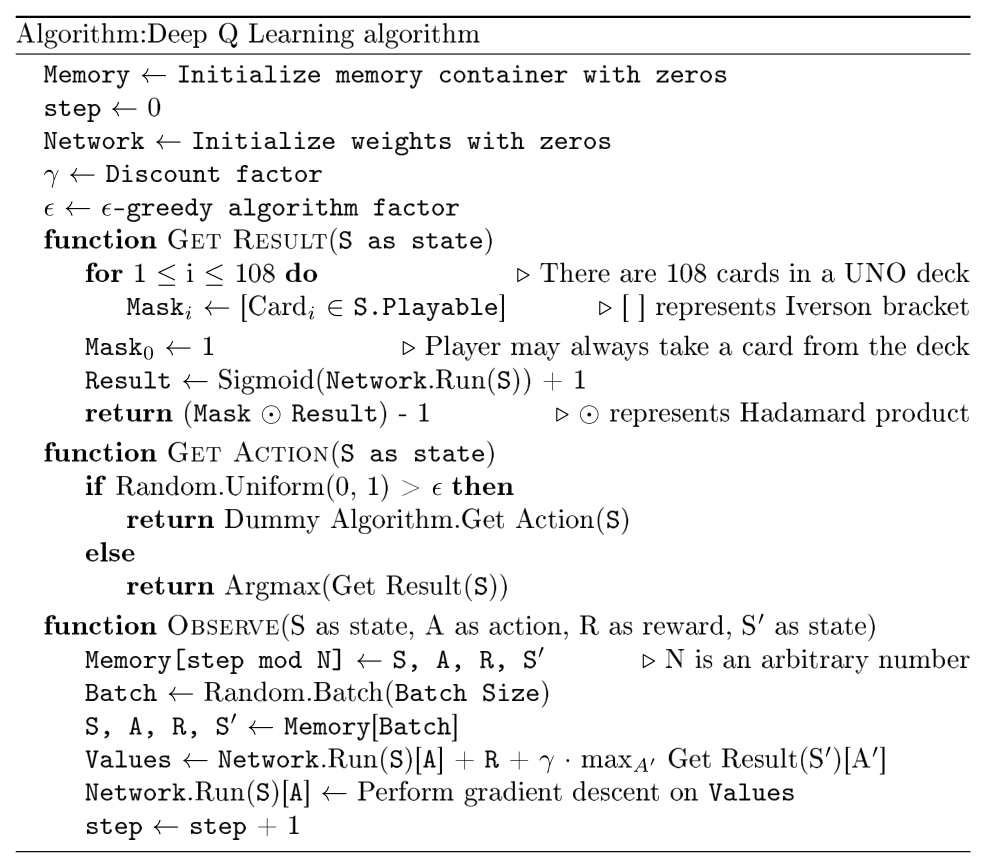
Defensive algorithm等同「先丟普通牌，再丟功能牌，沒牌再抽卡」，猶如杞人憂天，永遠把好牌留在手上，以備不時之需。

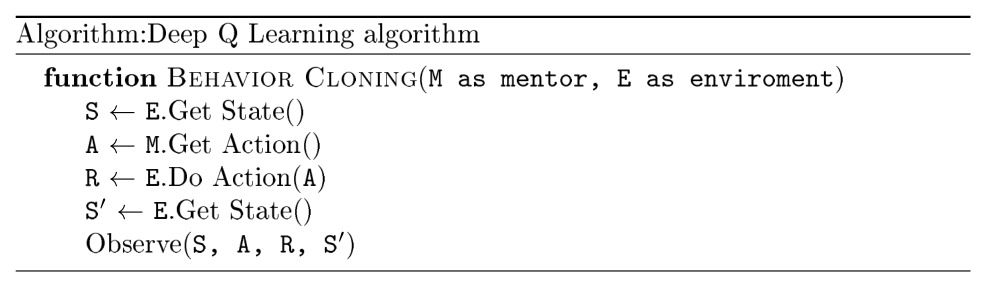
# 五、Genetic algorithm

演算法如同在一個培養皿(Petri dish)上模擬許多個Cells，細胞將與對手(Opponent)決鬥，根據決鬥結果計算細胞的最終積分(Rank)，再進行基因擴散(Genetic diffusion)，藉此交換基因。

不適應環境的基因會在基因擴散的過程中漸漸消失，適應環境的基因會在基因擴散的過程中壯大，如同達爾文的「適者生存」說，演化過程會淘汰掉不適應的個體，留下「贏家」，也就是適應環境的個體。

# 六、Deep-Q-Learning Algorithm

嚴格來說，玩UNO的過程不是Markov decision process，任何情況下，玩家都不能任意出牌。因此，我對演算法進行了一些調整，如下偽代碼所示。

Behavior cloning是一個常見的處理手法，如同徒弟模仿師傅的行為，能夠有效增加強化學習的成效。

陸、現階段研究成果

# 一、先手制霸

不論是使用哪一種演算法，先手的勝率都較高，如下表所示。

|  |  |
| --- | --- |
| 演算法 | 先手勝率 |
| Dummy algorithm | 53% |
| Naïve offensive algorithm | 52% |
| Naïve defensive algorithm | 56% |

# 二、Genetic Algorithm的特色 – 劇烈起伏

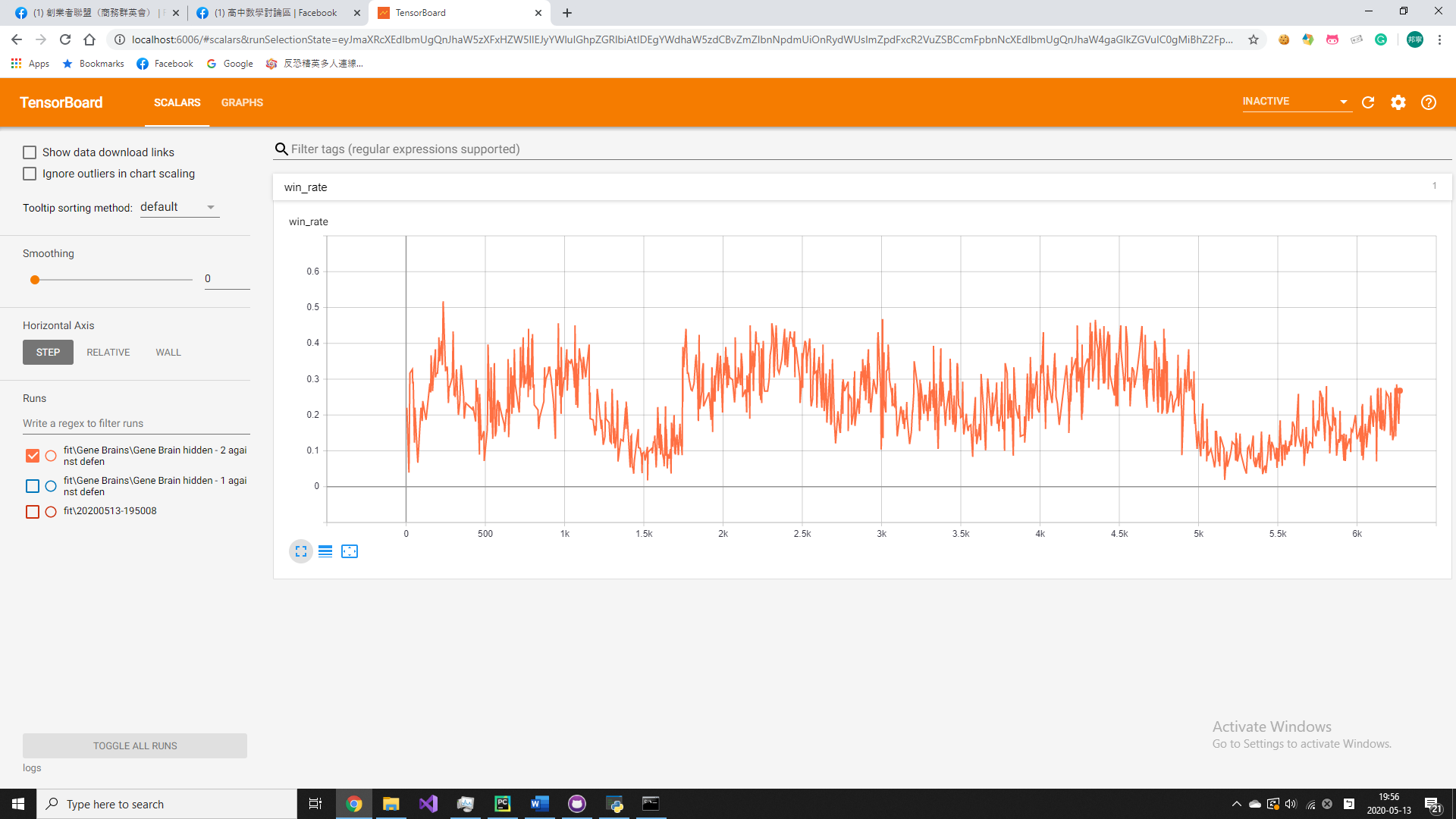
 可以見到Genetic algorithm的勝率時高時低，如下圖所示。

圖 3 ▲ Genetic algorithm對抗Defensive algorithm的勝率圖

好的基因，適應力強的基因，也就是勝率高的基因，會隨著基因擴散(Genetic Diffusion)快速的散播到培養皿(Petri)的每個細胞內，使得勝率增加。不過，在突變(Mutation)的影響下，好的基因也有可能變成壞的，就像惡性腫瘤一樣，使得勝率下降。

# 二、Deep-Q-Learning的特色 – 恍然大悟或瞬間失憶

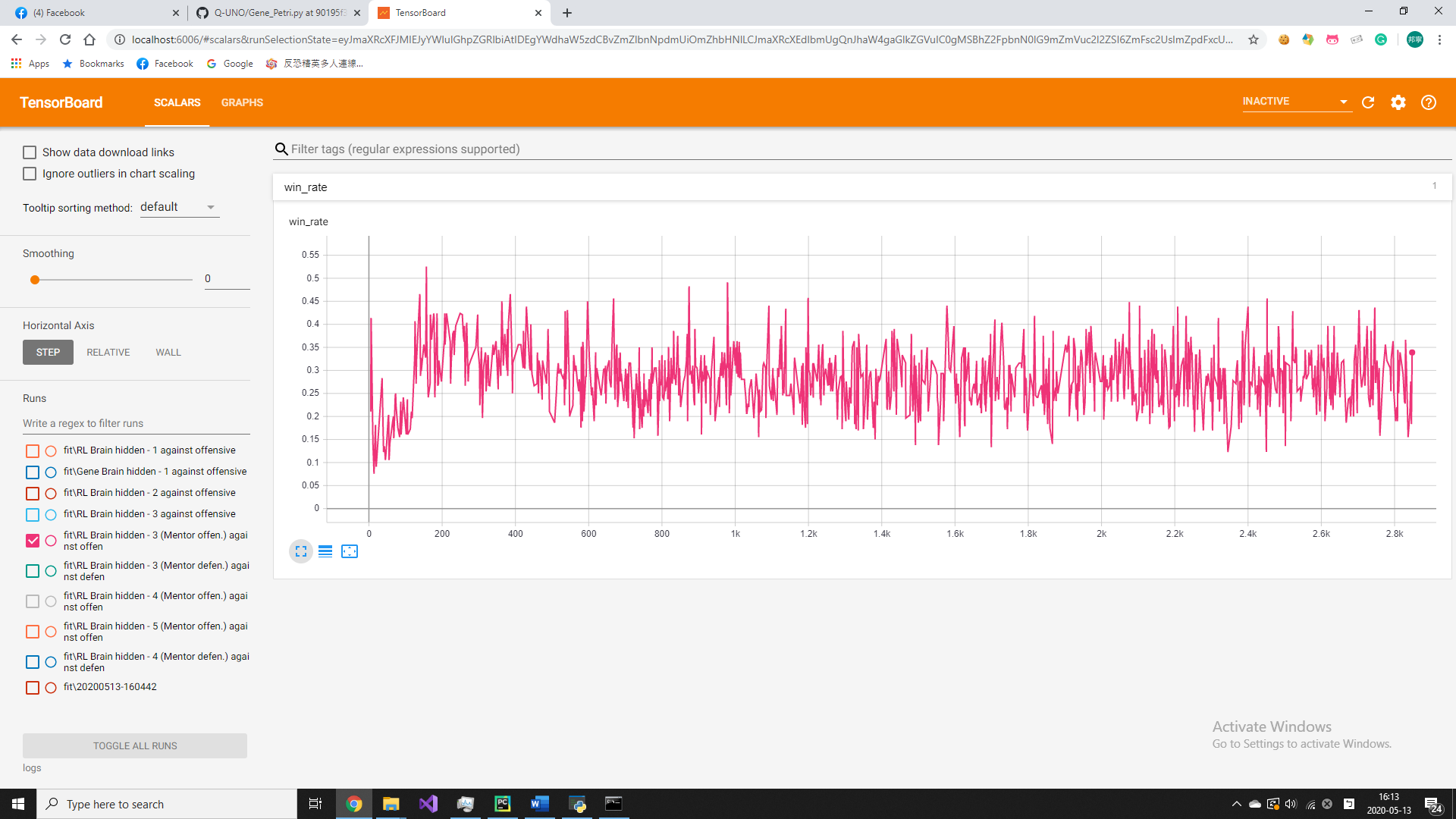
 可以見到，Deep-Q-Learning具有「恍然大悟」的潛力，神經網路彷彿突然觀察到某種重要線索，勝率瞬間飆升，如下圖所示。

圖 4 ▲ 3-Hidden Deep-Q-Learning搭配Behavior cloning(Offensive algorithm) 對抗Offensive algorithm的勝率圖

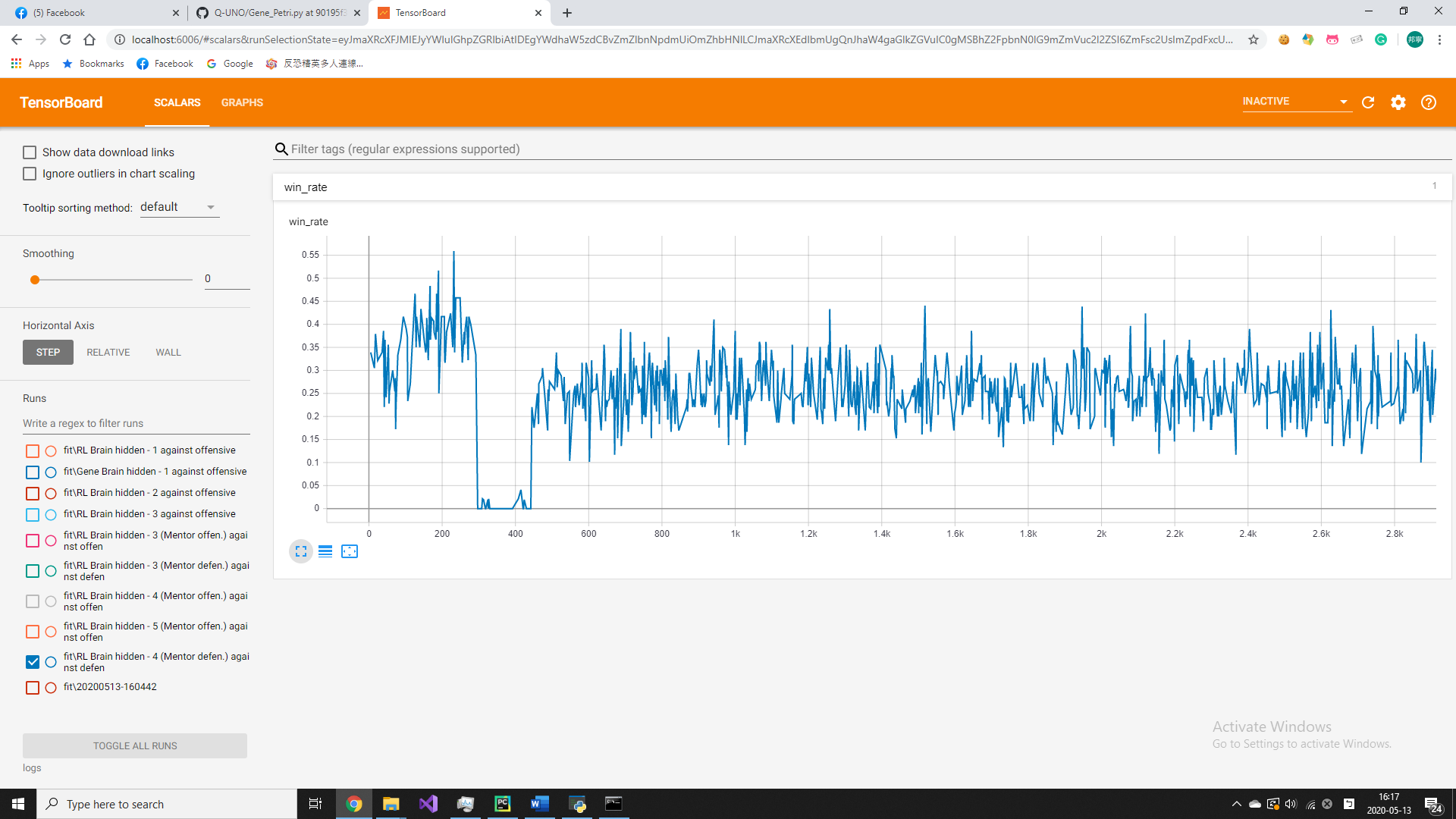
 相對的，Deep-Q-Learning也具有「瞬間失憶」的潛力，神經網路彷彿突然失憶，把畢生所學給全忘了，勝率瞬間跌落谷底，如下圖所示。

圖 5 ▲ 4-Hidden Deep-Q-Learning搭配Behavior cloning(Defensive algorithm) 對抗Defensive algorithm的勝率圖

由此可知，Deep-Q-Learning有可能「恍然大悟」，也有可能「瞬間失憶」，與Genetic-Algorithm相較，Deep-Q-Learning是較為穩定的演算法。

# 四、隱藏層深度

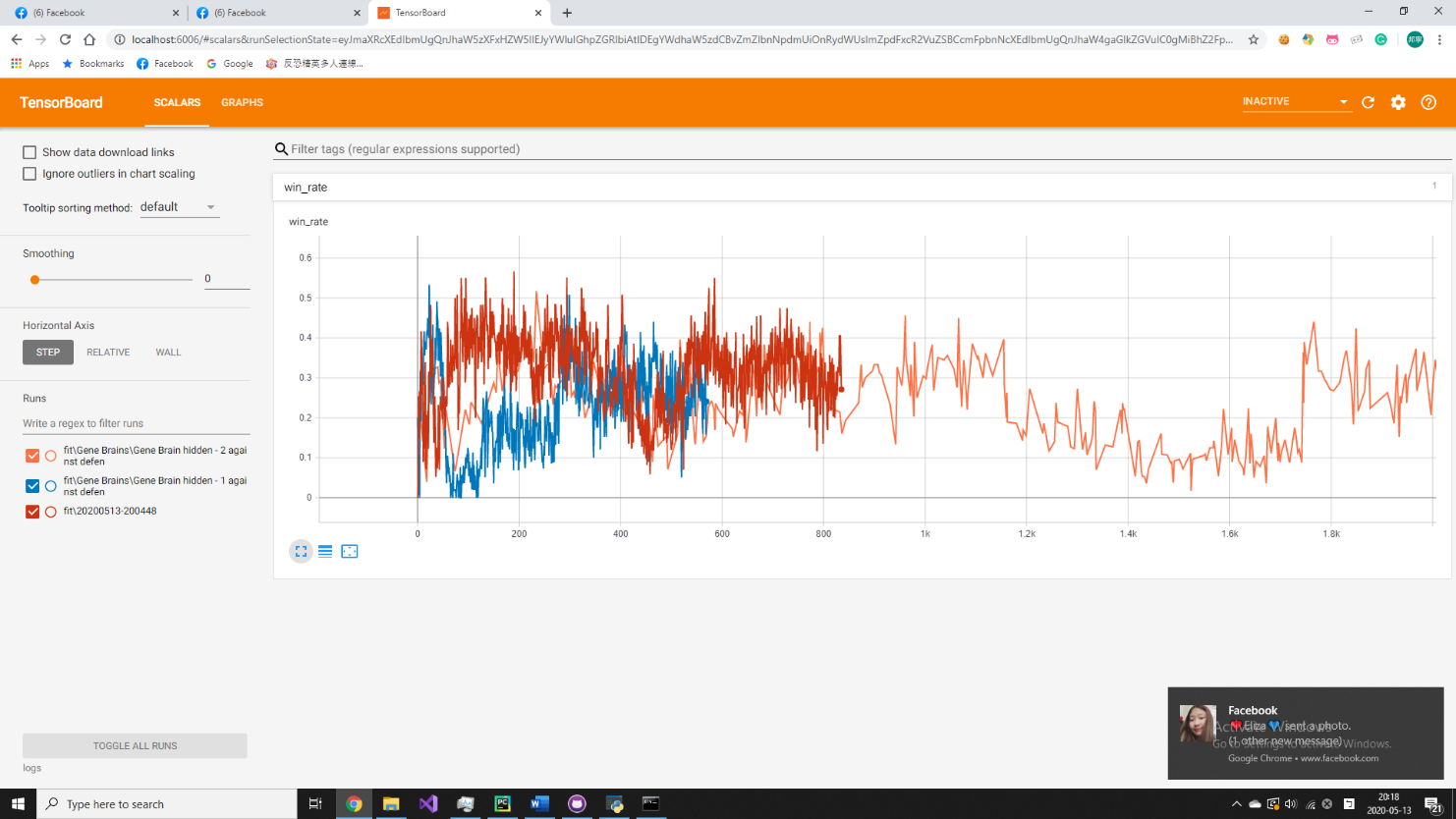
Genetic algorithm對隱藏層深度不太敏感，如下圖所示。

圖 6 ▲ Genetic algorithm對抗Defensive algorithm的勝率圖

|  |  |
| --- | --- |
| 顏色 | 意義 |
| 藍 | Genetic algorithm with 1-hidden layer |
| 橘 | Genetic algorithm with 2-hidden layer |
| 紅 | Genetic algorithm with 3-hidden layer |

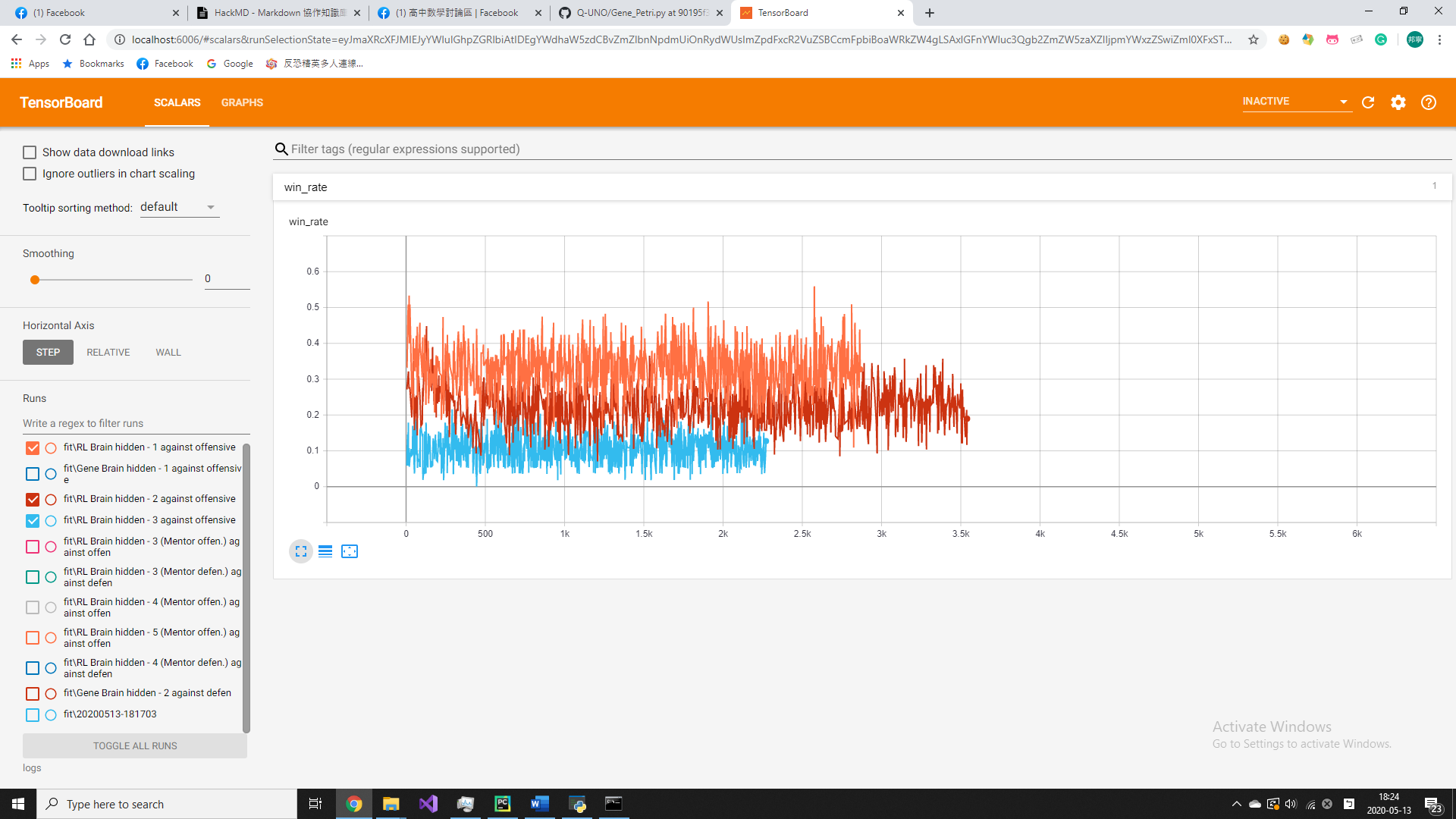
Deep-Q-Learning對隱藏層較為敏感，不使用Behavior cloning的話，隱藏層深度越深，反而會造成反效果，如下圖所示。

圖 7 ▲ Deep-Q-Learning對抗Offensive algorithm的勝率圖

|  |  |
| --- | --- |
| 顏色 | 意義 |
| 橘 | Deep-Q-Learning with 1-hidden layer |
| 紅 | Deep-Q-Learning with 2-hidden layer |
| 藍 | Deep-Q-Learning with 3-hidden layer |

深層的神經網路如塊璞玉，未經雕琢不得其瑰。若是環境提供的樣本品質不夠好，使得神經網路無法好好的被「雕琢」，便會有反效果；若是有一個師傅能夠提供高品質的樣本，去「雕琢」這塊璞玉，便能得到內部的瑰麗，便能得到更高的勝率。

Behavior cloning猶如師徒相授，師傅做什麼，徒弟便跟著做；過於淺層的神經網路如朽木，再怎麼教也教不會，神經網路擬和能力過差；過於深層的神經網路如大材小用，耗費大量運算資源，卻無優異成效。

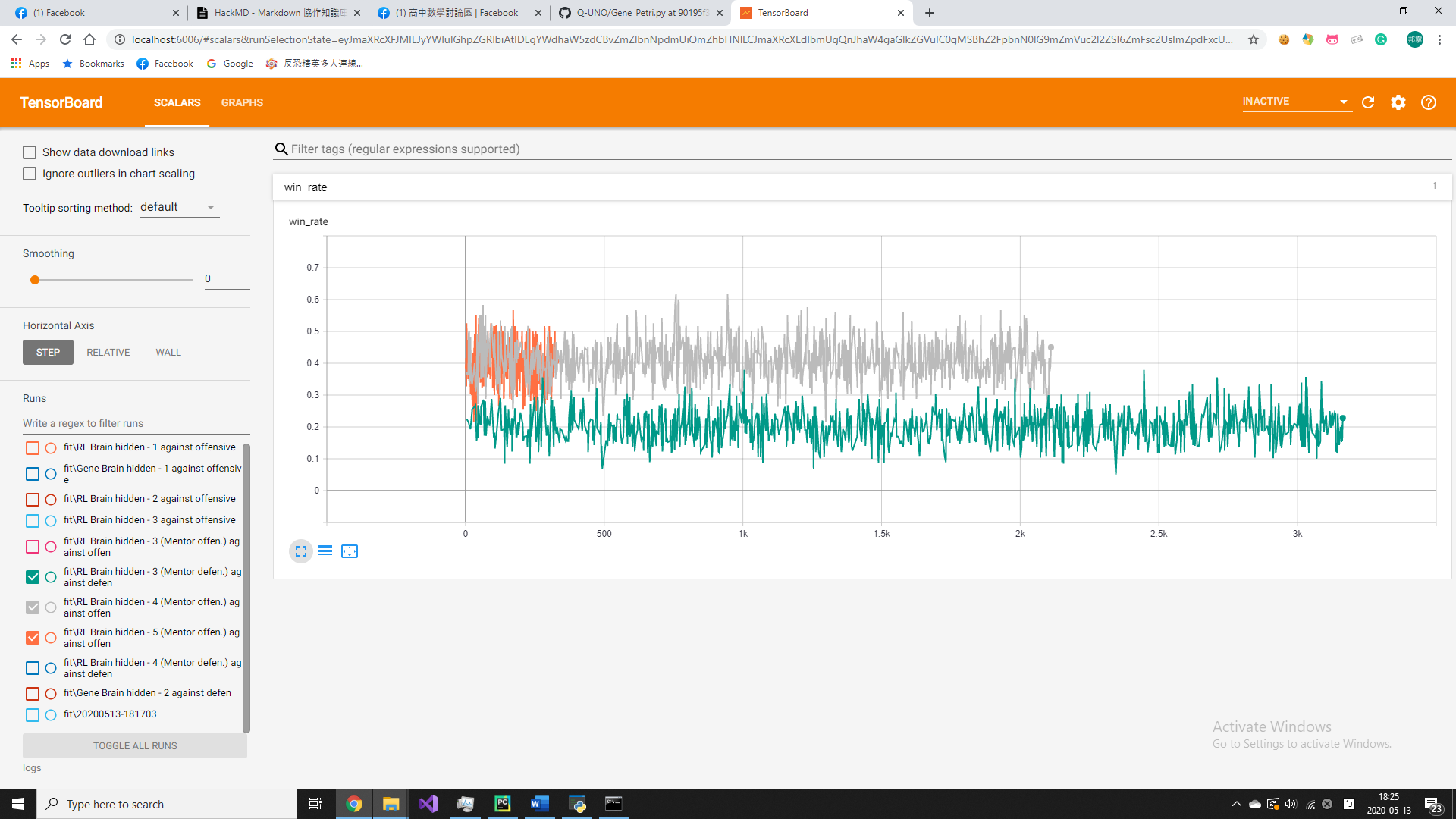
經過實驗歸納，筆者認為四層的神經網路搭配Behavior cloning，不僅能有效的增加勝率，也能妥善使用運算資源，最為恰當，因此，多數實驗都以此為標準。

圖 8 ▲ Deep-Q-Learning搭配Behavior cloning對抗Defensive algorithm的勝率圖

|  |  |
| --- | --- |
| 顏色 | 意義 |
| 橘 | Deep-Q-Learning with 5-hidden layer and offensive algo as mentor |
| 灰 | Deep-Q-Learning with 4-hidden layer and offensive algo as mentor |
| 綠 | Deep-Q-Learning with 3-hidden layer and offensive algo as mentor |

# 五、Genetic-algorithm對決Deep-Q-Learning

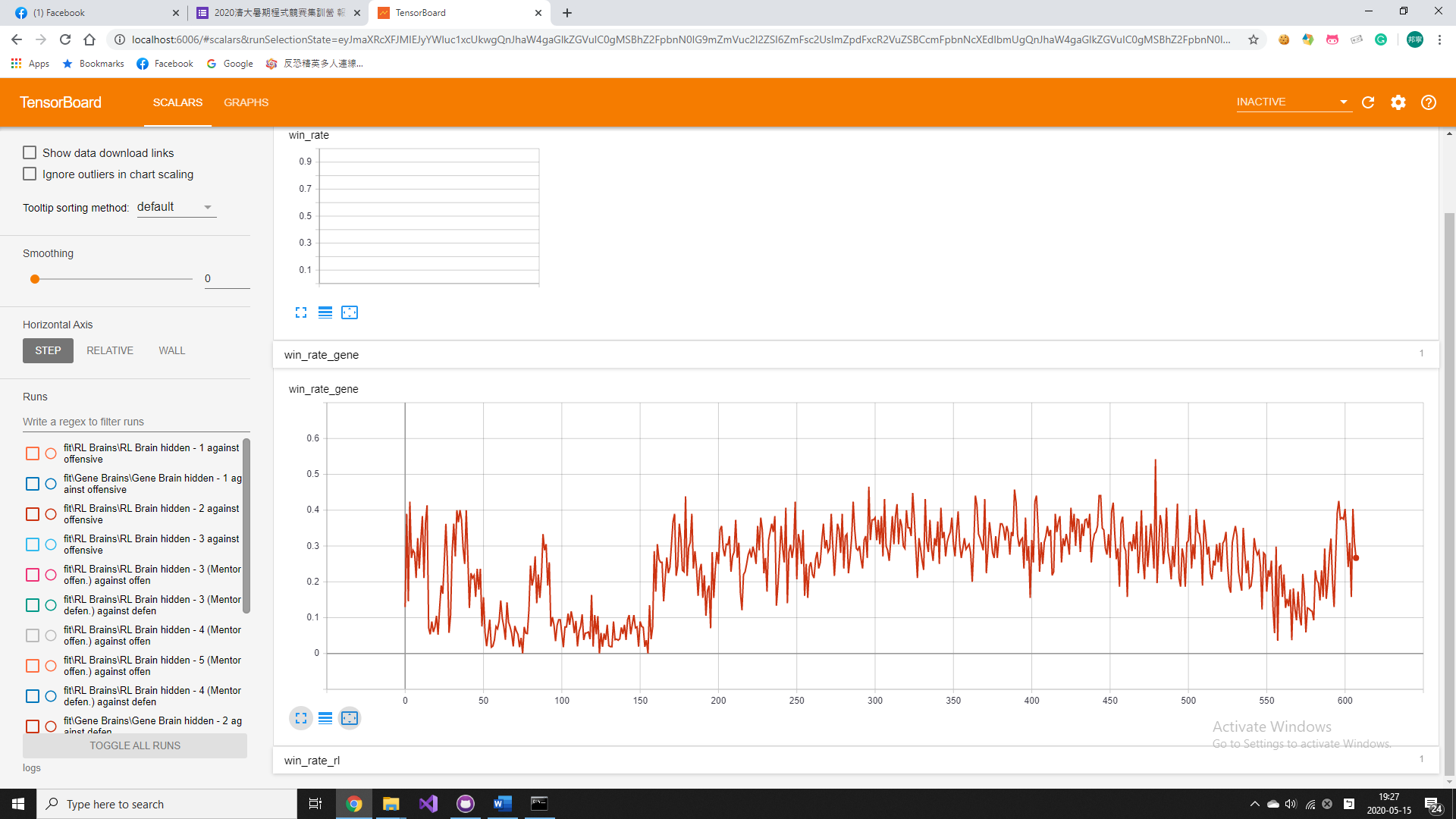
 Genetic algorithm好比細菌，Deep-Q-Learning好比人腦，兩者都是基於神經網路的演算法，而這場神經網路大戰，結果將會如何呢？下圖展示了Genetic algorithm對決Deep-Q-Learning的勝率。

圖 9 ▲ Genetic algorithm對抗4-Hidden Deep-Q-Learning搭配Behavior cloning(Defensive algorithm)的勝率圖

可以見到，多數時候Deep-Q-Learning佔上風，藉由Genetic fluctuation，Genetic algorithm有時能夠贏過Deep-Q-Learning，不過多數時候還是敗給Deep-Q-Learning。

|  |  |
| --- | --- |
| Gene algorithm勝率最低點 | Gene algorithm勝率最高點 |
| 圖 10 ▲ | 圖 11 ▲ |

在Deep-Q-Learning的全盛時期，勝率將近100%，可謂戰無不勝；在Genetic algorithm的全盛時期，勝率高達54%，可謂略勝Deep-Q-Learning一籌。

# 六、各演算法之間的勝率

下表展示了各演算法之間的勝率，並列出神經網路的重要規格、參數。

(一)、Genetic algorithm採用一層Hidden layer。

(二)、DQN採用四層Hidden layer，搭配Defensive algorithm作為師傅。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject  Object |  | Naïve  Offensive | Naïve  Defensive | Genetic  Algorithm | DQN |
| Dummy | 最高 | 93.8% | 93.1% | 58% | 96% |
| Naïve  Offensive | 最高 |  | 54.7% | 56% | 53% |
| Naïve  Defensive | 最高 |  |  | 53% | 57% |
| Genetic  algorithm | 最高 |  |  |  | 99% |
| 最低 | 46% |

可以觀察到幾個重點：

(一)、除了對上Genetic algorithm時，Dummy algorithm穩輸

(二)、Defensive algorithm比Offensive algorithm強

(三)、Deep-Q-Learning多數時候比Genetic algorithm好

(四)、不論是DQN或是Genetic algorithm都能與Naïve algorithm相抗衡

(五)、Genetic algorithm都只是略微勝出

柒、結論

# 一、探討先手與後手的利與弊

在特定條件下，先手能夠「完全制霸」對手，也就是存在一種出牌順序，使得先手必勝，如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 先手牌組 | 紅色停止 | 紅色迴轉 | 紅5 |
| 後手牌組 | 黑色+4 | 藍色+2 | Wild card |

先手只需要出「停止」、「迴轉」再出「紅5」，先手就贏了，若存在先手必勝策略，且先手是絕對理性的，後手將無機會逆轉，造成先手勝率較高。

# 二、探討各演算法之間的勝率

Naïve offensive algorithm對上Naïve defensive algorithm時，defensive有較高的機率勝出，這是因為offensive algorithm常常把功能牌先丟完了，使得最後沒有功能牌可以丟。根據筆者的自身經歷，多數人喜歡把功能牌留到最後一刻再丟；通常大家會傾向於「存功能牌」，並在遊戲結束前盡速丟出，就像Naïve defensive algorithm一樣。

因為Naïve algorithm是「死」的，而Deep-Q-Learning與Genetic-Network是「活」的，經過訓練後，神經網路能夠佔Naïve algorithm的小便宜。美中不足的是，UNO是個「資訊不對稱」的賽局，就算能夠掌握對手的行為模式，也沒辦法掌握對手的下一張牌，因為神經網路沒辦法看穿對手的牌，沒辦法像AlphaGo或是Deep Blue那樣「輾壓」對手。

Dummy algorithm遇到所有演算法幾乎都被「輾壓」，正如其名，他是設計來「被輾壓」的演算法，不論遇到誰都輸，名副其實的「對戰假人」。

# 三、探討Deep-Q-Network與Genetic-Network的利與弊

如果把Deep-Q-Network譬喻成人類，而Genetic-Network譬喻成細菌的話，我們可以用相當有趣的角度來看待這場遊戲。

細菌使用基因作為記憶，酵素作為武器，向人類宣戰；原始的人類也使用基因作為記憶，免疫系統作為武器，向細菌宣戰；神農氏跨出了偉大的一步，使用大腦作為記憶，草藥作為武器，向細菌宣戰；微生物學家使用文字作為記憶，抗體、抗生素以及噬菌體作為武器，向細菌宣戰。

在這數千年的歷史中，人類不斷進步，發明新的武器與細菌對戰；在這數千年的歷史中，細菌也不斷進步，演化出新的基因與人類對戰，這不就跟電腦裡發生的事情一模一樣嗎？Deep-Q-Learning不斷想出新主意去對付Genetic algorithm，Genetic algorithm不斷演化出新策略去對付Deep-Q-Learning，好比人類研發出新疫苗對抗細菌，細菌演化出新偽裝去對抗人類！

Genetic algorithm時常劇烈起伏，好比細菌的攻擊能力時強時弱；Deep-Q-Learning比Genetic algorithm穩定許多，好比人類懂得使用語言、文字以及記憶來儲存所學所能，凡是學會的，就會永遠保存在在腦中。

下表簡潔有力的表達了兩種演算法之間的利弊得失，以及之間的關係。

|  |  |  |
| --- | --- | --- |
|  | Genetic algorithm | Deep-Q-Learning |
| 優勢 | 神經網路構造簡單 | 勝率較為穩定  搭配Behavior cloning效果顯著 |
| 劣勢 | 增加隱藏層深度成效不彰  勝率十分不穩定 | 神經網路構造複雜 |
| 好比 | 細菌 | 人類 |

# 四、未來展望

在目前所完成的研究之外，還有不少研究工作十分值得探討，以下列舉三點。

## (一)、基於作夢啟發的Deep-Q-Learning

人類在睡眠期間常常作夢，作夢的內容大多為近期發生的事件，科學家相信這是腦袋正在進行記憶回放(Memory playback)，並進行學習。若是將「作夢」的概念應用在Deep-Q-Learning上面呢？經過了一天的努力工作，給予Deep-Q-Network一個空檔，讓他好好作夢，讓他回放(Playback)記憶，進行學習，會不會有更好的成效呢？

根據網路文獻回顧，沒有人提出相關的論文，也沒有人進行相關實驗，筆者認為，作夢算法能夠有效利用隱藏層，使得勝率再更進一步的提高。

## (二)、改變神經網路的輸入格式

有一句俗話「當你解不開這題，不如換個角度去看這題」，改變神經網路的輸入格式，就好比換個角度看世界。笛卡爾座標就是「換個角度看世界」的經典例子，有些數學問題難以用代數工具解開，卻能輕易的被幾何工具解開；若是神經網路能夠跳脫代數的框架，用幾何的角度思考，便能解開複雜的問題，便能更進一步的提高勝率。

## (三)、網頁遊戲

目前人機對戰的介面相當的不友善，若是想要進行人機對戰實驗，則必須要有一個更友善的介面。筆者預計用Unity Web作為使用者介面，Tensorflow + Python + Nodejs作為後台，將這篇研究做成一個網頁遊戲，方便進行人機對戰實驗。

# 五、總結

昔日，Deep Blue使用自己強大的運算效能，窮舉每一種狀況，最終攻克了西洋棋領域；近年，Alpha Go使用卷積神經網路以及Q-Learning，演算哪個落點勝率最高，終於攻克了圍棋領域。

本研究設計了諸多演算法，經過實驗後得知，這些參賽者中的黑馬是Deep-Q-Learning，不僅勝率穩定，還能夠藉由Behavior cloning增加勝率，是目前最強的演算法。

繼Deep Blue、Alpha Go之後，電腦又將矛頭指向全世界最暢銷的桌遊──UNO，蓄勢待發，準備攻克這個領域！

捌、參考資料

|  |  |
| --- | --- |
| 參考資料 | 連結 |
| UNO on wiki | <https://zh.wikipedia.org/wiki/UNO#%E5%85%A9%E4%BA%BA%E5%B0%8D%E7%8E%A9%E7%9A%84%E8%A6%8F%E5%89%87> |
| Deep-Q-Learning paper | <https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf> |
| Deep-Q-Learning tutorial | <https://www.mlq.ai/deep-reinforcement-learning-q-learning/> |
| Genetic algorithm on wiki | <https://en.wikipedia.org/wiki/Genetic_algorithm> |
| RL cards | <https://arxiv.org/pdf/1910.04376.pdf> |
| Q-Learning on wiki | <https://en.wikipedia.org/wiki/Q-learning> |
| Tensorflow | <https://www.tensorflow.org/api_docs/python> |
| Github repository | <https://github.com/lawrence910426/Q-UNO> |