Gomoku Deep

[1] 李佳陽,[2] 王泰淞,[3] 張騏岳,[4] 江尚紘,[5] 曾士桓

[12345] 國立高雄科技大學電腦與通訊工程系

[1]E-mail: 0551082@nkust.edu.tw

[2]E-mail: 0551084@nkust.edu.tw

[3]E-mail: 0551088@nkust.edu.tw

[4]E-mail: 0551080@nkust.edu.tw

[5]E-mail: shtseng@nkust.edu.tw

摘要

本篇論文將 Self-play reinforcement learning 方法實作在五子棋對弈，其方法包含了 Monte-Carlo Tree Search(MCTS)演算法以及深度學習的技術，透過自我訓練進而不斷強化人工智慧在五子棋對弈的能力，以便達到能與人進行對戰的程度。

訓練完成後，與人進行對戰方面，玩家一開始可以選擇先手以及後手，接著選擇要遊玩的難易度，總共分為 Easy、Normal 和 Hard，讓玩家可以與不同強度的人工智慧進行對弈，並體驗其中不同強度人工智慧的差異。

關鍵字: MCTS；Self-play reinforcement learning；Deep neural network。

1.前言

從 1950 年被圖靈提出的「圖靈測試」中得知，如果這台機器能通過設備能與人進行對話且被誤認為人，那麼這台機器就具有智能，一直到達特茅斯會議中才被正式提出人工智慧這詞，也是到這個時候人工智慧這一詞才被真正的定義，往後便開啟了 AI 的時代，從數學定理的證明到投入工業生產，雖然中途也有不少學者提出反對研究 AI 的主張，或是不受到政府機構的支持，但是到了現在，由 AI 發起的革命還是相當成功，現金的部分機器人、影像辨識、市場分析，甚至是 Google 的搜尋引擎，皆能發現 AI 人工智慧的影子。

隨著 AI 人工智慧的大時代來臨，許多企業紛紛投入 AI 相關技術，以為獲取更大利益或減少成本開銷，不需花費大量人力資源便可獲得相同甚至更好的成果，所以在現今社會中所需的便是具有 AI 方面相關知識的人才，不僅僅是因為目前是一個處處皆需要大數據的時代，而是要能夠有效處理運用這些大數據資料的方式，光依靠以前的方法很難有效達到現今所要求的標準及速度，所以就得依靠經由人工所訓練出的強大智能體，在資料處理分析等等用途上，不管是效率還是速度皆遠遠是人所處碰不及的。

1.1 研究目的

在前些日子看見有關 AlphaGo 以圍棋的方式打敗各個圍棋好手之後，開始對於人工智慧這塊領域產生了興趣，但由於圍棋的規則掌控和要訓練一輪所需花費的時間過於龐大，於是選擇了另一種棋類活動，也就是五子棋，來實踐人工智慧中機器學習與深度學習這個部分。

2. 文獻探討

此論文講述關於 AlphaGo 整個系統的架構，從策略模擬至自我對下的過程皆有詳細描述，主要架構是由深卷積神經網路加上對局資料庫進行訓練，透過 Monte-Carlo Tree Search 強化整個 Policy Network，再利用 Policy Network 以迴歸方式計算出 Value Network，這樣在實際對弈時不僅可以使用 Policy Network 搭配 Monte-Carlo Tree Search 尋選擇落子位置，並使用 Value Network 縮短 Monte-Carlo Tree Search 的時間及搜尋深度且預測輸贏的

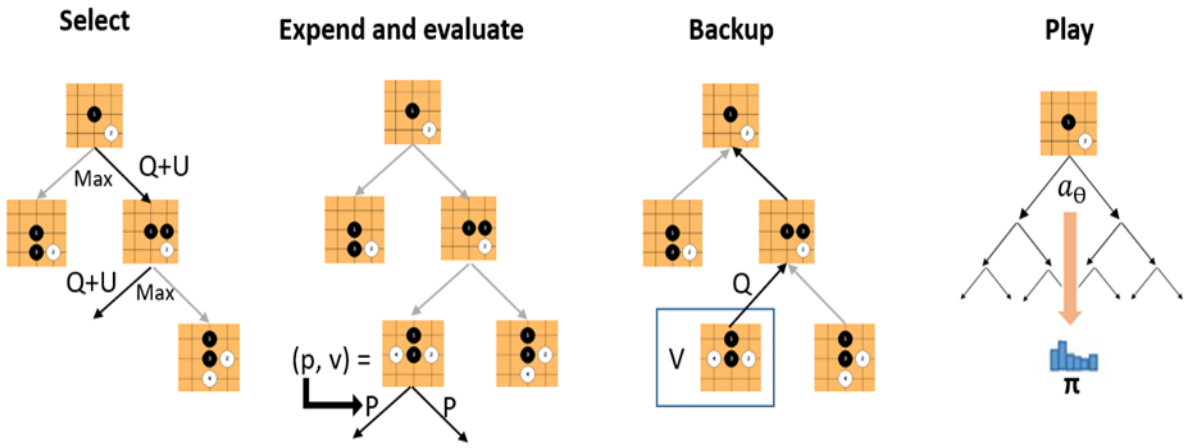

圖 1 Monte-Carlo tree search[1]

結果，而在實際對弈中也確實地打敗了世界棋王[7]。

根據此篇論文內容，主要描述 AlphaGo Zero 在學習過程中，不需要依靠人對它進行指導，也就是不必藉由吸收我們提供的知識及經驗，透過此篇描述的新的強化學習演算法，使用一個無經驗的神經網路搭配搜尋演算法，靠著自我對弈，從零逐漸提升為超越 AlphaGo 的水準[1]。

兩者最大的差異在於 AlphaGo 在訓練時需要依靠棋譜，人為處理過的數據，也就是過往的經驗並進行反饋來預測或決策，而 AlphaGo Zero 卻不用，只需要給他圍棋的規則，讓其從自我對弈中不斷的摸索，憑藉著自我訓練來建立出自己的模式，時間上，AlphaGo Zero 也遠遠少於 AlphaGo，所需用的硬體資源也較少，這也能看出，利用監督學習加上強化學習與單純的強化學習這兩者所展現出的結果。

相較於這兩篇論文提到的演算法，雖然一開始也使用過 DQN 建構的架構來進行五子棋的訓練，但其展現出的強度卻不如我們所預期，對於整體完整性來說，不僅是強度有些不足，更是容易達到收斂而就算繼續訓練也難以提高其強度，而這兩篇提到的方法對於使用普通電腦進行訓練則需要花費大量時間，畢竟此搜尋演算法必須不斷地深入與重複，但也因為這個強大的搜尋演算法才能將效能及品質提升一個檔次。

3. Self-play reinforcement learning 研究方法

3.1 Monte-Carlo Tree Search(MCTS)結合深度學習

圖 1 為 MCTS 的流程，其說明如下：

Select 始於樹之根節點，當遇到葉節點則終止，選擇落子是透過具有最大值的 Q(s, a) 加上 upper confidence bound(UCB) U(s, a) 來決定，而 U(s, a) 取決於先前儲存的概率 P 以及該節點的訪問次數 N，N 隨著訪問的次數遞增。

Expend and evaluate 如遇到葉節點則把當前 s 輸入 Deep Neural network 得出該節點之(p, v) 訊息，接著進行 Backup。**Backup** 從葉節點開始回傳該路徑之 Q 值與 V 值。**Play** 一旦搜尋完成，會在節點 s 處選擇 a 進行落子。

對於整體來說，MCTS 是外層的大框架，Deep Neural network 則是輔助 MCTS 進行搜索。因此對於一個盤面 s，我們使用 MCTS 建造一棵樹，而在建造的過程中若碰到葉節點則返回其(p, v)。建立完成後則會取得π，並根據π來決定下一步棋的位置，接著取得下一個 s，重複此過程，一局棋結束後則會取得結果 z。

3.2 策略價值網路(Policy-Value Network) 訓練

在最一開始，由於 model 未經過訓練，因此在 self-play 的過程中，就如同兩個皆不會下棋的人在對弈，也因此要用很多時間去訓練和優化，而在訓練過程中，要得知 model 是否有優化與更強，就必須每隔一段時間去評估它。

從圖 2 的自我訓練流程圖中，我們與 AlphaGo Zero 最大差別是在評估神經網路這步驟，AlphaGo

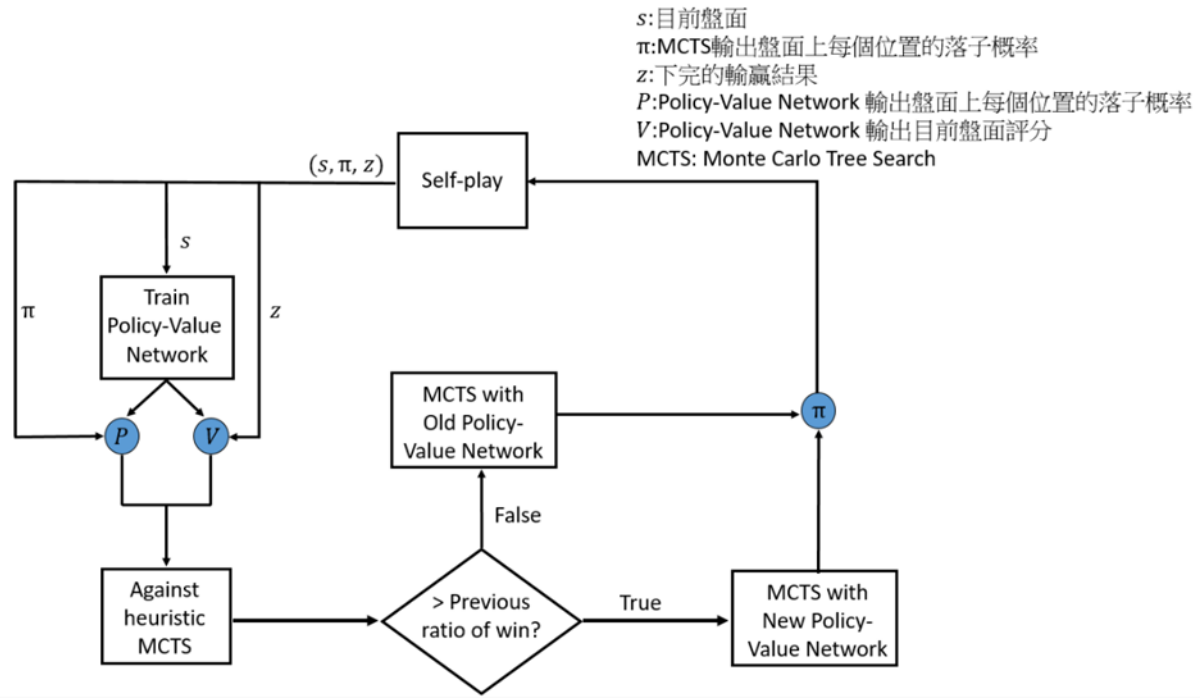

圖 2 五子棋對弈之自我訓練流程圖

Zero 用的評估方式是每隔一段時間，將新的 model 和舊的 model 進行對戰，有如自己每訓練一段時間去跟一個和自己實力相當的人對戰，不過這方法的缺點較難以判定自己是否真的變強，而且 model 收斂速度較慢；而為了要在短時間訓練成靠譜的 model，所以，我們改變了評估方式，就是每隔一段時間，將新的 model 和加入 heuristic function 的 Pure MCTS 去對戰，這就像自己每訓練一段時間，去跟一個會下棋的人對戰，看自己的實力是否有提升，以及提升了多少。綜合上述，跟 AlphaGo Zero 相較之下，我們所用的評估方式可讓 model 收斂速度有較快的優點。

我們 Training 的方法是利用 Self-play 搭配強化式學習的演算法，去訓練策略價值網路(Policy Value Network) [1]。詳細說明如下：

a. 訓練的 data 從哪裡來?

我們訓練的 data 是透過 Self-play，如圖 3 所示，Self-play 的 data 直接從當前最佳的 model 產生，並用來優化自己。

b. Self-play 生成 Data 的多樣性

為了訓練出有效的策略價值網路，就需覆蓋各式各樣的局面，所以 data 的多樣性是相當重要的，因此我們在 self-play 中加入了 2 種 exploration 的手段，每一次 self-play 前 30 步，使用 MCTS 搭配策略價值網路去 exploration，之後的 exploration

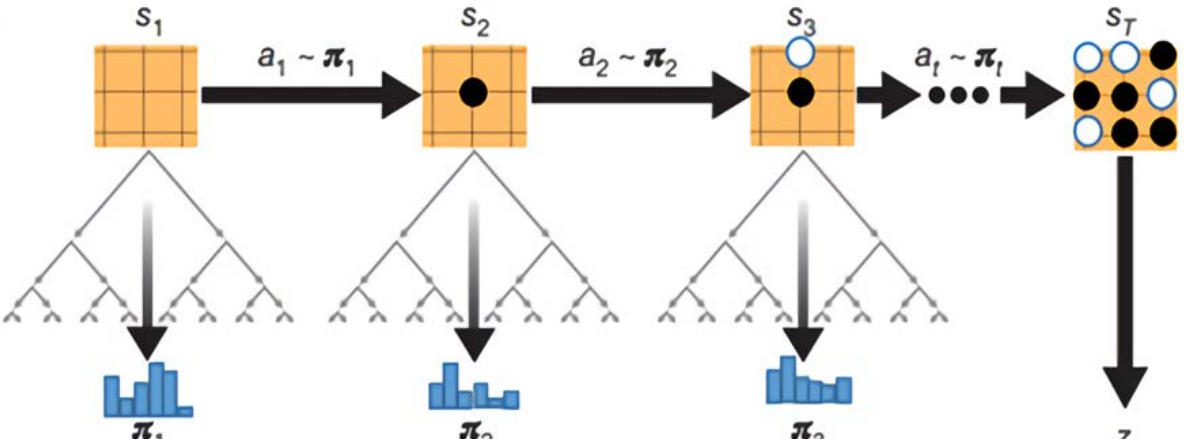

圖 3 Self-play 流程圖［1］

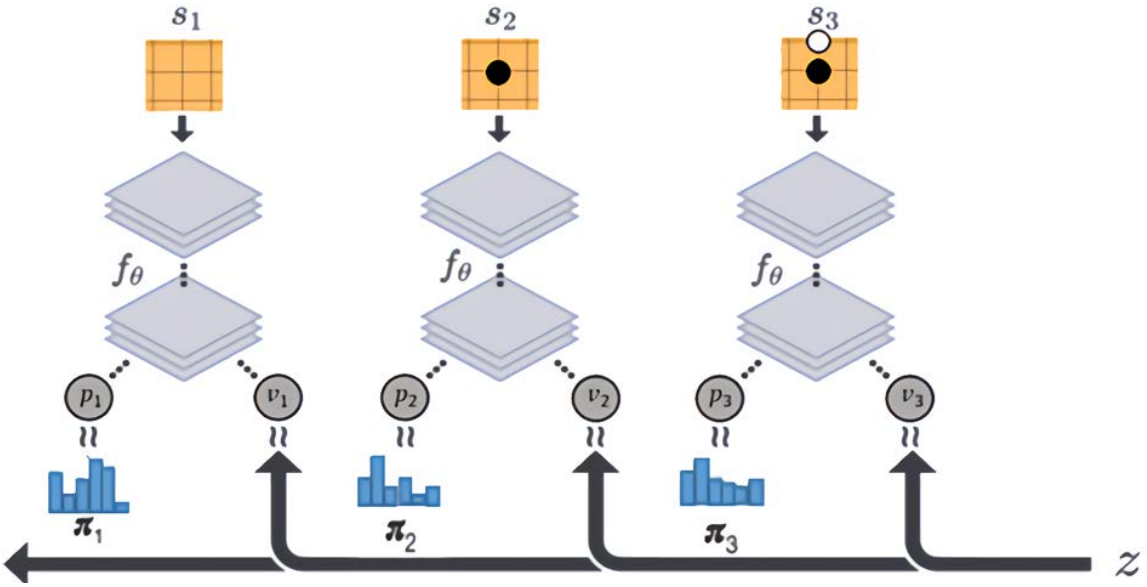

圖 4 策略價值網路訓練架構圖［1］

是用 Dirichlet noise 去實現 [6]。

c. Data 的保存和擴充

在 self-play 過程中，我們會以當前 player 去儲存一些數據 (s,π,z) 到 data buffer 中，其中 s 代表盤面，會用兩個二值矩陣分別表示自己和對手棋子的位置，另外，五子棋也和圍棋一樣有對稱性的等價性質，所以我們會把 s 進行旋轉和鏡像翻轉，擴充成 8 種等價狀況；π 是 MCTS 輸出的概率，z 是目前這局下完的結果，贏等於 1，輸等於 -1，平手等於 0 [6]。

d. 何謂策略價值網路

所謂的策略價值網路 f_θ ，就是給定它當前的盤面 s 作為輸入，而輸出當前盤面每個位置下棋的機率 p 和一個對於整個盤面評分的值 v ，$(p,v) = f_\theta(s)$ [1]。

e. 策略價值網路架構

圖 4 顯示當一局棋結束後則會取得結果 z 之後，針對每一步中的策略價值網路預測出的v_t進行最佳化。單一的策略價值網路如圖 5 所示，第一層到第三層使用卷積層，分別使用 32、64、128 個 3x3 的 filter，使用 ReLu 激勵函數。然後分成 policy 和 value 這兩個輸出，policy 這端使用 4 個 1x1 的 filter 來降維，再連接一個全連接層，最後透過 Softmax 非線性函數直接輸出棋盤上每個位置的落子概率；value 這端先使用 2 個 1x1 的 filter 來降維，然後連接一個 64 個神經元的全連接層，再連接一個全連接層，最後使用 tanh 非線性函數直接輸出 [-1,1] 之間的局面評分 [6]。

f. 策略價值網路訓練

根據策略價值網路訓練過程圖，我們的目標就是要讓策略價值網路輸出的概率 p 更接近 MCTS 輸出概率 π；讓輸出盤面的的評分 v 更接近預測實際結果 z。優化方面，我們參考 AlphaGo Zero 的論文使用的損失函數:

$$l = (z-v)^2 - \boldsymbol{\pi}^T log\boldsymbol{p} + c\|\theta\|^2 \quad [1] \quad (1)$$

其中第三項是用於防止過擬合的正規項 [6]。

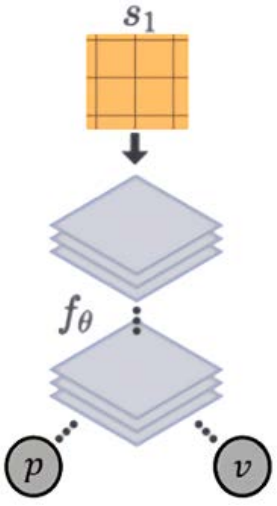

圖 5 策略價值網路 [1]

g. 策略價值網路的評估方式

每隔 50 次的 self-play，我們會對當前的 model 進行評估，評估方式是將新的 model 搭配 MCTS 和加入 heuristic function 的 Pure MCTS AI 去對戰 10 局，其中 Pure MCTS AI 最開始每下一步棋使用 1000 次模擬，當我們訓練的 AI 以 10 : 0 結束時，就把 Pure MCTS AI 每下一步棋模擬的次數加 1000 次，增加 Pure MCTS AI 的強度，就這樣依此類推，不斷增強。

h. 整體訓練流程

根據前面幾點所提的，self-play 所蒐集的 data 就是為了訓練策略價值網路，而評估完後，最優的 model 也會馬上被用在 MCTS 來進行 self-play，生成更好的 self-play data，就這樣兩者互相促進，構成訓練的循環 [6]。

4.實驗結果

圖 6 展示的是一次在 9*9 棋盤上進行五子棋訓練的過程中損失函數隨著 self-play 局數變化的情況，這次實驗一共進行了 3000 局自我對局，損失函數從最開始的 4 點多慢慢減小到了 2 點多左右。[6]

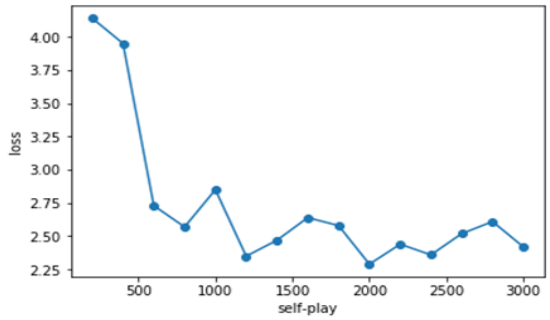

圖 6 loss 局數變化值

除了觀察到損失函數在慢慢減小，我們還觀察策略價值網絡輸出的策略（輸出的落子概率分佈）的 entropy 的變化情況，如圖 7。正常來講，最開始的時候，我們的策略價值網絡基本上是均勻的隨機輸出落子的概率，所以 entropy 會比較大。隨著訓練過程的慢慢推進，策略價值網路會慢慢學會在不同的局面下哪些位置應該有更大的落子概率，也就是說落子概率的分佈不再均勻，會有比較強的偏向，這樣 entropy 就會變小。[6]

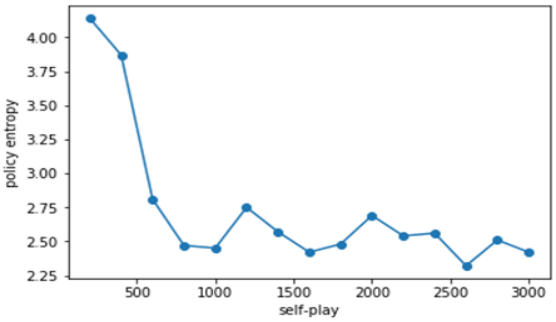

圖 7 policy 的 entropy 變化

5.結論與討論

本篇論文是用 Monte-Carlo tree search 搭配 Policy-Value Network 實現 Self-play reinforcement learning 方法，並應用在五子棋對弈上。一開始我們是使用 DQN 架構實作，但是其結果皆不理想，我們認為是提供 Training 的 data 太少，加上 Reward 設計無法達到完美，導致最終無法收斂。因此最終改成使用本發表之方法實作，此方法的成果較為理想。

Gomoku Deep 對於棋面邊角的位置較缺乏對抗性，也就是說玩家一直下邊角位置即會獲勝；這部分將於 Training 進行改進與修改，期盼能將此電腦訓練成：在棋面的任何角落，均能達到專業水準。

6.參考文獻

[1] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis,‘‘ Mastering the Game of Go without Human Knowledge’’, *Nature* ,2017

[2] tobe,「如何學習蒙特卡羅樹搜索(MCTS) 」，Tensorflow 專欄，106 年 11 月 2 日

[3] Yeg James，「卷積神經網路介紹(Convolutional Neural Network)」，第 5.1 講，Medium，106 年 12 月 24 日

[4] Rachel Liao，「Reinforcement Learning 進階

篇:Deep Q-Learning」，Medium，107 年 10 月 2 日

[5] 莉森揪，「Deep Q Network(DQN)之術是解析」，iT 邦幫忙，AI&Date，第 25 篇，107 年 11 月 9 日

[6] 宋俊瀟，AlphaZero 實戰:從零學下五子棋，106 年 12 月 24 日

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre,George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman,Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu,Thore Graepel & Demis Hassabis ‘‘Mastering the game of Go with deep neural networks and tree search’’, *Nature*, 2016