**國立東華大學資訊工程系**

**National Dong Hwa University**

**112學年度大學部畢業專題研究報告**

**112 CSIE Undergraduate Project Report**

**愛因斯坦棋之人工智能開發**



|  |  |
| --- | --- |
| **指導教授Advisor：** | **顏士淨教授** |
| **專題參與人員Team Member：** | **張繼元** |
|  | **林源毅** |
|  | **李俊諺** |

中 華 民 國　　113　 　年　　5　　月 　24 　 日

**國立東華大學資訊工程學系**

**專題報告原創性聲明**

**National Dong Hwa University**

**Department of Computer Science and Information Engineering**

**Statement of Originality**

本人鄭重聲明：

所呈交的專題報告是在指導老師指導下進行的研究工作及取得的研究成果。除文中已經註明引用的內容外，本報告不包含任何其他個人或集體已經發表或撰寫過的研究成果。對本文的研究做出重要貢獻的個人與集體，均已在文中以明確方式標明。若有違上述聲明，願依校規處分及承擔法律責任。

I hereby affirm that the submitted project report is the result of research under the supervision of my advisor. Except where due references are made, the report contains no material previously published or written by another person or group. All significant facilitators to the project have been mentioned explicitly. Should any part of the statement were breached, I am subject to the punishment enforced by the University and any legal responsibility incurred.



|  |  |  |
| --- | --- | --- |
| 學號Student No. | 學生姓名Name | 親筆簽名Signature |
| 411021212 | 張繼元 |  |
| 411021236 | 林源毅 |  |
| 411021228 | 李俊諺 | IMG_256 |



日 期: Date 2024/5/24

**計畫摘要**

Keywords : 愛因斯坦棋, 蒙地卡羅樹搜尋, 深度學習, bitboard

愛因斯坦棋是2004年才被發明的新興棋類，雙方在大小為5x5的棋盤上進行骰棋遊戲。為了提升愛因斯坦棋的棋力，我們利用深度學習中的卷積神經網路，經過資料前處理、建立模型、模型訓練等一系列過程，將訓練好的模型與蒙地卡羅樹搜尋做結合，並改變原本資料結構為bitboard，以其位元運算的特性加速程式。

**目錄**

[第一章 前言 1](#_Toc22914)

[第二章 研究動機與研究問題 2](#_Toc9975)

[2.1 研究動機 2](#_Toc1836)

[2.2 研究問題 2](#_Toc6399)

[第三章 相關文獻 3](#_Toc26533)

[3.1 蒙地卡羅樹搜尋 3](#_Toc23812)

[第四章 研究方法與步驟 5](#_Toc20450)

[4.1 深度學習 5](#_Toc22913)

[4.1.1 資料前處理 5](#_Toc17515)

[4.1.2 建立模型 6](#_Toc8461)

[4.1.3 模型訓練 7](#_Toc21752)

[4.1.4 與蒙地卡羅樹搜尋結合 8](#_Toc13811)

[4.2 修改UCT 9](#_Toc20025)

[4.2.1 Progressive Bias 9](#_Toc32627)

[4.3 改良模擬階段 9](#_Toc14999)

[4.3.1 新模擬策略 9](#_Toc9705)

[4.4 Bitboard 9](#_Toc12839)

[第五章 實驗結果與討論 13](#_Toc30078)

[第六章 結論與未來展望 15](#_Toc15157)

[參考文獻 16](#_Toc8972)

# 第一章 前言

愛因斯坦棋(德語:EinStein würfelt nicht!)，是在2004年由德國中部耶拿鎮數學教授Ingo Althöfer所發明的兩人骰棋類遊戲，與棋類遊戲中常見的圍棋、象棋、五子棋等相比，是一種非常新穎而且鮮為人知的棋類。

遊戲初始：

棋盤大小為5x5，如圖1-1，雙方皆擁有編號1至6的棋子，隨機擺放在自己陣地角落的六個位置，先、後手可以用任何方式決定。



**圖1-1** 初始盤面

走步與吃子：

雙方輪流擲一個六面骰，骰到的點數即為可移動的棋子編號，若骰到的點數對應的棋子已不在場上，則從大於或小於該點數最近的棋子擇一進行走步，例如：擲出3，若3不在場上，則2、4號棋皆為可動子，擇一進行走步。藍方走步時，可移動的方向為左、上或左上，紅方則為右、下或右下，可自行選擇。吃子方面需要注意的是，若移動的目的地有棋子的話，則不論敵我該棋都會被吃掉，移出棋盤。

勝利條件：

1. 其中一方吃光對面所有棋子。
2. 其中一方任一棋子走到對方的角落，即紅方棋子走到右下角，藍方棋子走到左上角。

# 第二章 研究動機與研究問題

## 2.1 研究動機

2016年3月，DeepMind團隊所開發的AlphaGo以4:1的成績戰勝韓國職業圍棋棋士李世乭九段，這次對弈證明了人工智慧在電腦對局方面的成功，激發我們對此領域的興趣，朝向這方面做研究。

與圍棋棋盤大小19x19相比，愛因斯坦棋棋盤大小只有5x5，複雜度低很多，不過愛因斯坦棋有骰子這個不確定因素影響著棋局，也因為比較新穎，對此棋的研究較少，還有許多可以研究的地方，經由教授的建議以及組別內的討論，我們決定將愛因斯坦棋作為我們本次研究。

## 2.2 研究問題

本次研究的目的，在於利用蒙地卡羅樹搜尋演算法的基礎上，設計bitboard的資料結構保存盤面以加快計算速度，並且利用深度學習卷積神經網路的技術，在蒙地卡羅樹的節點選擇上更具準確性，提高下棋程式的棋力。

**第三章 相關文獻**

**3.1 蒙地卡羅樹搜尋**

蒙地卡羅樹搜尋(Monte Carlo tree search；簡稱：MCTS)，是一種使用蒙地卡羅演算法在遊戲樹中搜尋的方式。

蒙地卡羅演算法，也稱類比統計法，是指使用亂數來解決很多計算問題的方法，以機率為基礎，通過隨機抽樣的方法，以隨機事件出現的頻率估計其機率，或者以抽樣的數字特徵估算隨機變數的數字特徵，模擬的次數越多，越接近實際上的機率，結果也會越精確。

蒙地卡羅樹搜尋的每個迴圈包括四個步驟：

1. 選擇（Selection）: 從根節點*R*開始，連續向下選擇子節點至一個尚未擴充的節點，也就是葉節點L。選擇節點的方式以及考量因素有很多，通常都會與勝率以及探索程度有關，如何向最佳的方向擴充，是設計蒙地卡羅樹搜尋的重點。

2. 擴充（Expansion）: 除非任意一方的輸贏使得遊戲在L結束，否則建立一個或多個子節點並選取其中一個節點*C*。

3. 模擬（Simulation）: 再從節點*C*開始，用隨機策略進行遊戲，又稱為playout或者rollout，如何調整策略，使的遊戲更貼近真實情況，也是蒙地卡羅搜尋樹中的重要一環。

4. 反向傳播（Backpropagation）: 使用隨機遊戲的結果，更新從*C*到*R*的路徑上的節點資訊。

一張含有 圓形, 圖表, 行, 設計 的圖片

自動產生的描述 一張含有 圓形, 寫生, 圖表, 圖畫 的圖片

自動產生的描述

**圖3.1-1** 選擇（Selection）  **圖3.1-2** 擴充（Expansion）

一張含有 圓形, 寫生, 圖畫, 圖表 的圖片

自動產生的描述 一張含有 圓形, 寫生, 圖畫, 圖表 的圖片

自動產生的描述

**圖3.1-3** 模擬（Simulation） **圖3.1-4** 反向傳播（Backpropagation）

# 第四章 研究方法與步驟

## 4.1 深度學習

深度學習是機器學習的分支，是一種以人工神經網路為架構，對資料進行表徵學習的演算法。由於近年大數據的普及為深度學習提供了充足的訓練數據，以及硬體設備進步使得計算能力提升，讓深度學習模型訓練更準確、更高效，能夠有效地應用在各個領域。

### 4.1.1 資料前處理

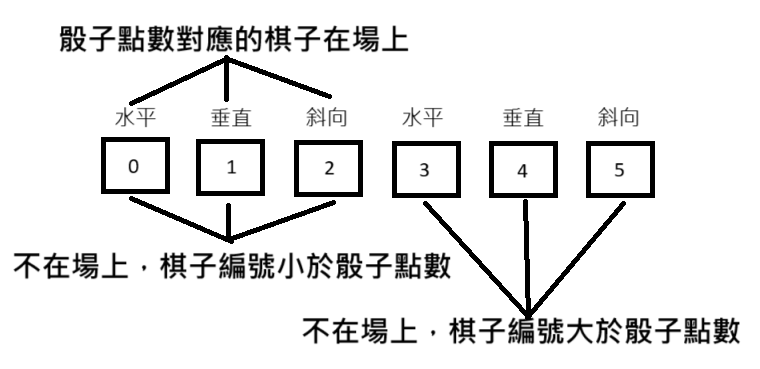
資料從Little Golem(有各種棋類遊戲的平台)上取得top 10 players的棋譜，並將資料中重複的棋局、尚未下完的棋局從數據集裡移除，最後得到85226個棋局，總共有1447890筆資料，儲存成CSV檔。

將棋盤視作一張圖片，每一個位置視為圖片的pixel，因此一個feature map的大小為5x5的二維陣列。

本研究總共用了16個feature map，如下所示:

1. 6個feature map各別為紅方編號1~6棋子的所在位置為1，其他為0。
2. 6個feature map各別為藍方編號1~6棋子的所在位置為1，其他為0。
3. 1個feature map為可動子的所在位置為1，其他為0。
4. 1個feature map全1。
5. 1個feature map全0。
6. 1個feature map為空格(沒放雙方棋子)的位置為1，其他為0。

資料標記的部分，考慮一回合最多6個走步，因此將每一筆資料標記為數字0~5。如果骰子點數所對應的棋子在場上的話，只使用0~2分別為水平、垂直、斜向三個方向的走步；如果對應的棋子不在場上的話，0~2為滿足棋子編號小於骰子點數、最接近骰子點數與存在場上這三個條件的棋子，分別為水平、垂直、斜向的走步，3~5與0~2類似，只差在3~5是棋子編號大於骰子點數的狀況。



**圖4.1.1-1**資料標記圖解

### 4.1.2 建立模型

使用Pytorch為開發框架，Colab為開發環境，建立一個卷積神經網路(Convolutional Neural Network, CNN)。

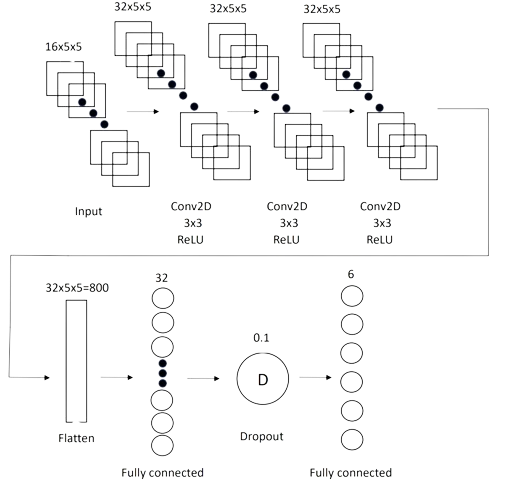
其網路輸入為16個feature map，接著經過三層convolutional layer，其卷積核大小皆為3x3，且皆產生32個feature map。

卷積運算的方式為從左上角開始，從左到右，由上到下，每次移動一格，對當前卷積核所包覆的區域做內積運算，直到右下角結束。每次卷積運算後，使用激活函數(activation function)增加神經網路的非線性，這裡使用Rectified Linear Unit(ReLU) function，將小於0的值改變為0，大於等於0的值則維持不變。

經過三層convolutional layer後，先經過一層flatten layer，將目前所有的feature map展開成一維的，再連接一層fully connected layer，輸出32個神經元。

下一層為dropout layer，此層只在訓練時才發揮作用，其功能為防止過擬合(overfitting)，設定一個參數，其參數表示要隨機捨棄多少比例的神經元。

最後輸出層是fully connected layer，輸出6個值(神經元)，這6個值代表一回合中最多6個走步的各個機率。

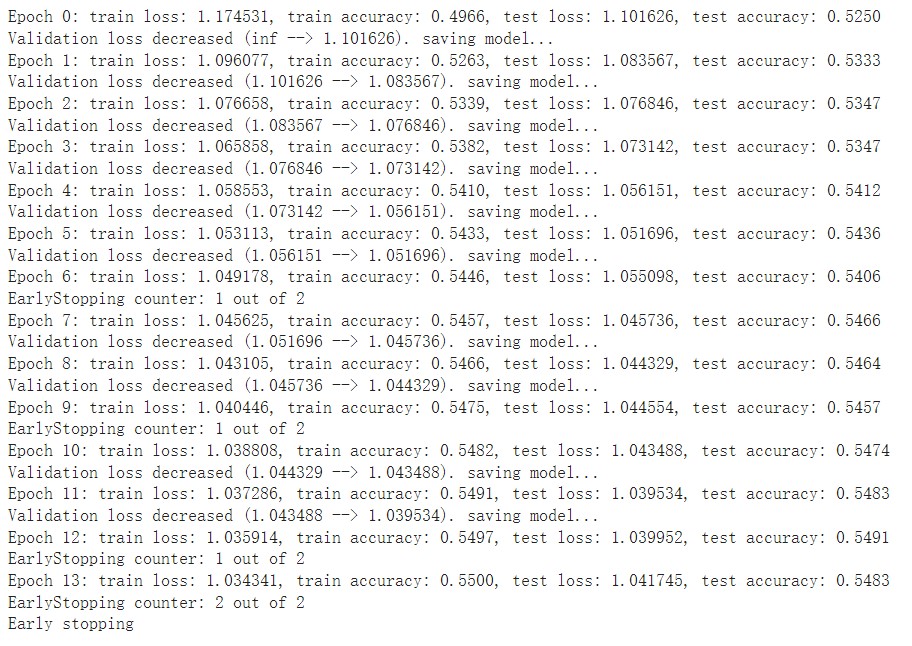


**圖4.1.2-1**類神經網路架構

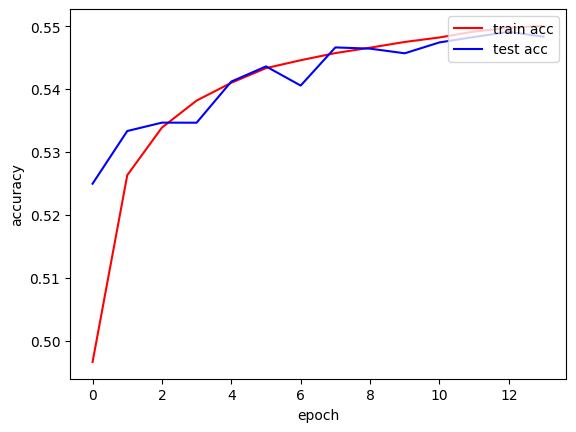
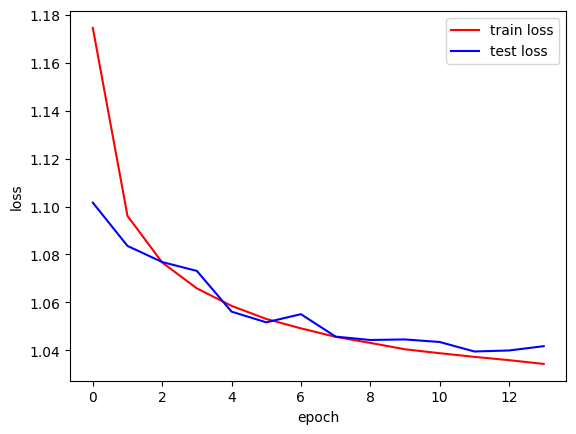
### 4.1.3 模型訓練

在每個epoch中，訓練集與測試集的數據都是從原始數據集裡面隨機抽取的，訓練集佔90%、測試集佔10%，隨機抽取的好處是模型能夠在各個epoch中看到不同的訓練資料，因而提高模型的泛化能力。

此外，在每個epoch的最後，加入了早停(Early Stopping)機制，判斷如果連續2個epoch測試的損失值都比測試最佳的損失值還大的話，就停止訓練，而會在測試損失值最佳的時候儲存模型



**圖4.1.3-1**模型訓練損失值及準確率



**圖4.1.3-2**訓練與測試損失值 **圖4.1.3-3**訓練與測試準確率

隨epoch變化趨勢 隨epoch變化趨勢

### 4.1.4 與蒙地卡羅樹搜尋結合

在選擇階段時，把當下的棋盤狀態以及該節點的點數及位移方向通過特徵擷取成模型所需要的16個對應特徵並傳入模型，再把模型輸出棋子合法步之概率值儲存在該節點。

另外，在選擇階段，把該值增加進UCT公式内，進行節點選擇。

## 4.2 修改UCT

### 4.2.1 Progressive Bias

在選擇階段時，進一步修改原UCT (Upper Confidence Bound Applied to Trees) 公式。爲了增進節點選擇的準確度，我們對原本MCTS在選擇階段所使用的UCT公式增加了Progressive Bias的應用，公式如下：

其中 𝐶𝑃 為常數，𝐻𝑏 是透過預先訓練所得到當前盤面之平均勝率，𝑉𝑛 則是節點的拜訪次數。此方法稱為Progressive Bias。利用這個方法能夠直接將預先訓練所得到的知識直接加入選點的策略之中並增進節點選擇的準確度。

修改之後的UCT公式，使得原本的搜尋樹除了對表現較為良好的節點進行更加深入的探索外，同時也能夠適當地給予表現雖然不是最好但具有潛力的節點更多機會。修改掉了原本只搜尋最具有潛力的節點，避免了MCTS算法陷入局部最優解。

## 4.3 改良模擬階段

### 4.3.1 新模擬策略

在模擬階段時，原模擬的策略如下：

1：收集盤面上所有存在的棋子數

2：計算出我方棋子所有的合法步

3：如果存在合法步，隨機選擇走步

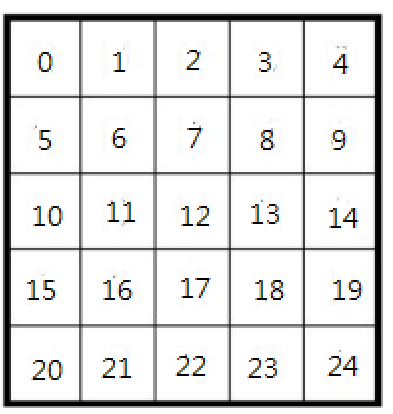
4：更換選手，直到一方勝利

爲了增加盤面的準確率，新增加了

5:當存在其中一個合法步是決勝手，將優先被選擇。

## 4.4 Bitboard

Bitboard是一種儲存資料的形式，由於比起處理陣列，電腦處理位元的效率更高，因此設計了一種特殊的資料結構，將原先5x5的棋盤變成一個長度為25的位元(bit)，並且透過不同的位元運算來表示盤面的移動和狀態。



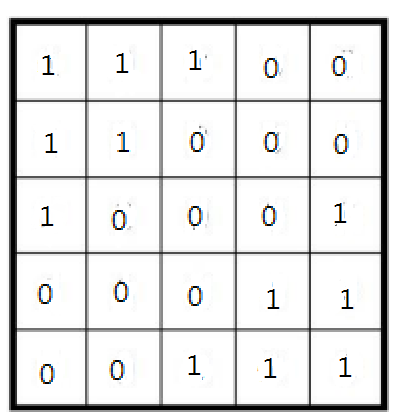
**圖4.4-1**每一位元對應的棋盤格子

宣告一種資料型態為32 bit的整數，但只使用其中25 bit，每一個bit都表示一格棋盤上的位置，由於愛因斯坦棋的走步與吃子只關乎棋子與棋子之間的相對位置，棋子與棋子之間並無大小關係，因此在進行走步的更新時，不需要考慮多餘的可能性，因此實現bitboard運算的過程較為簡單。

為了表示一個完整的盤面，總共需要13張bitboard，也就是13串長度為25的位元串:

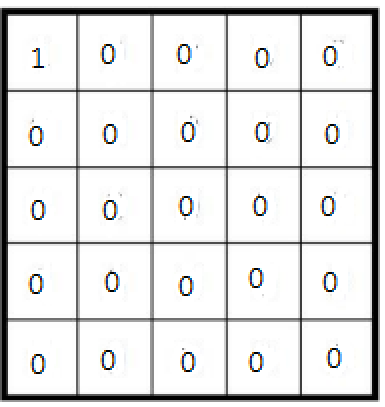
1. 全圖bitboard(1個)：上方有棋子的地方為1沒有棋子的地方為0。
2. 各棋子所在位置的bitboard(12個)：在該棋子所在位置上設為1其餘位置皆為0。

舉個例子，遊戲開局時的全圖bitboard為1110011000100010001100111。

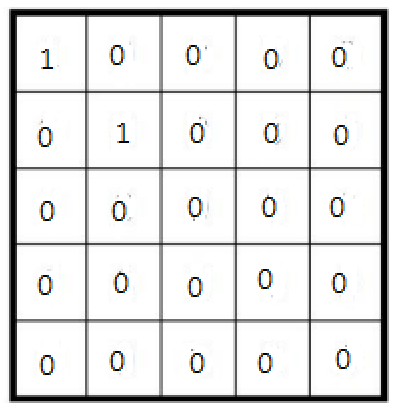


**圖4.4-2**開局時的全圖bitboard

對於bitboard來說，走步運算是一件很迅速的事情，例如：要將目前在最左上角的棋子向右下移動一格。

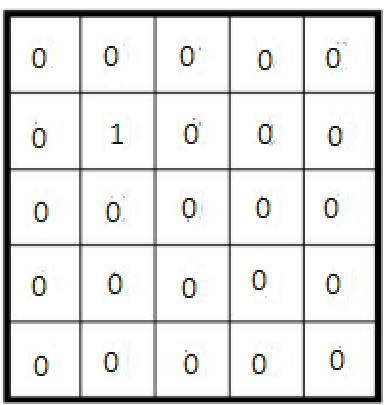


**圖4.4-3**原先盤面的bitboard為：0000000000000000000000001



**圖4.4-4**將原先棋子的位置以及將要移動到的位置都設為1的bitboard為：0000000000000000001000001

將這兩者做xor運算，即可得到更新後的盤面。



**圖4.4-5**執行走步完後的盤面

在此基礎上就可以做到一局遊戲內的走步與吃子，並且，要判斷勝負時只需確認藍方是否有棋子的bitboard為0000000000000000000000001，或是紅方是否有棋子的bitboard為1000000000000000000000000，這可以判斷是否有玩家走到終點，亦或是確認是否有一方的棋子bitboard全為0，代表該玩家的棋子全被吃光了。

# 實驗結果與討論

**對手程式命名為：MCTS**，其使用之策略為蒙地卡羅樹搜尋，模擬節點選用隨機走步的方式進行，選擇階段為UCT公式進行選擇。

**修改之程式命名：MODEL**，其使用之策略為蒙地卡羅樹搜尋，並改變其模擬和選擇階段，以及增加深度學習。

|  |  | 賽局一 | 賽局二 | 賽局三 |
| --- | --- | --- | --- | --- |
| MODEL贏局數 | 先手 | 278(55.6%) | 332(66.4%) | 366(73.2%) |
| 後手 | 222(44.4%) | 309(61.8%) | 313(62.6%) |
| 總勝率 | 50% | 64.1% | 67.9% |

**表5**每一場次分別對局1000場，並分爲先手後手記錄

賽局一：MODEL使用與MCTS相同的策略，對局1000場。

賽局二：MODEL使用與MCTS相同的策略，並加入深度學習及修改UCT公式為progressive bias，對局1000場。

賽局三：MODEL使用與MCTS不相同的simulation策略，並加入深度學習及修改uct公式為progressive bias，對局1000場。

|  |  | 賽局四 | 賽局五 |
| --- | --- | --- | --- |
| MCTS/MODEL贏局數 | 先手 | 442(88.4%) | 459(91.8%) |
| 後手 | 403(80.6%) | 441(88.2%) |
| 總勝率 | 84.5% | 90% |

**表5-1** MCTS與MODEL分別與randomAi對局1000場，並分爲先手後手記錄

賽局四：MCTS 與 randomAi對戰

賽局五：MODEL 與 randomAi對戰

由以上測試結果可觀察到，加入深度學習並優化模擬階段策略的程式勝率提升了17.9%。而其中先手勝率相較與後手勝率又高出了許多。此可説明在愛因斯坦棋中，因其先手會比後手多走一步，所以先手的勝率值會比較高。而因其擁有的隨機性較高，因此勝率值並不能做到完美的100%，只能盡可能地提高其勝率值。

賽局四和賽局五分別是讓兩個程式同時與randomAi互打，經過觀察，可以發現修改後的MODEL程式之勝率會比MCTS高了5.5%。

# 第六章 結論與未來展望

為提升愛因斯坦棋程式的棋力，我們利用深度學習與蒙地卡羅樹搜尋的結合來優化愛因斯坦棋程式，我們針對蒙地卡羅樹搜尋策略上的改良，搜尋的階段使用深度學習訓練好的模型，加入UCT中，並對其更改成Progress Bias的公式，模擬階段，爲了確保其隨機性，保留了原隨機走步的策略，並增加了判斷決勝手，減少其走多餘步的概率。為清楚了解優化程度和改進方向，進而開發了愛因斯坦棋自動對局程式，其中可清楚顯示牌面狀況與勝率。以兩個愛因斯坦棋程式MCTS與MODEL進行對局，我們以MODEL的身分參與對局，進而優化MODEL的下棋策略。

MODEL優化後的結果為，在1000場與MCTS對局之中，MODEL贏679場，優化後的MODEL勝率為67.9%。

因愛因斯坦棋中的隨機性，對其場次3的數據進行多了3000場的對局，通過處理得出最終的先手勝率值70%+-3，以及後手勝率值為62.5%+-2，總勝率值為66.25%+-3。

未來若要進一步進行愛因斯坦棋程式策略優化和程式加速，可分為兩個方向:

(1). 加強深度學習：因爲目前的模型勝率還有待加強，可對其網路層及資料庫方面進行更多的改良以達到更高的準確率。另外，由於目前的深度學習只是利用資料庫内的資料進行訓練，所以易忽略一些棋局，因此也可通過加入強化式學習，讓AI可以自行優化其準確率。

(2). 建立完整殘局庫：爲了加速愛因斯坦棋程式，所以有了殘局庫的構想，所謂殘局庫是利用雙方剩餘的手牌，直接搜尋勝負的結果。將預測結果加入殘局庫，往後在模擬時可直接利用盤面，便可得知結果。

# 參考文獻

1. 曹少剛(2017)，深度學習用於愛因斯坦棋研發之初步探討，國立臺灣師範大學資工所碩士論文。
2. 陳志昌(2022)，Bitboard 與 N-Tuple Networks 於愛因斯坦棋之應用，國立臺北大學資訊工程學系碩士論文。
3. Source code來自<https://github.com/tyhsu/Ewn> ，加以修改。