國立臺南大學資訊工程學系畢業專題

基於強化學習之格鬥遊戲即時反應操作 AI 設計

AI of Fighting Game

編　　號　：NUTN-CSIE-PRJ-109-020
執 行 期 間　：108 年 02 月 20 日至 109 年 01 月 09 日
專題參與人員：黃仁鴻
指 導 老 師　：陳榮銘

一、摘要

一般格鬥遊戲上，常有入門門檻過高與玩家間常出現嚴重的技術斷層等等問題，新手玩家連初級 AI 都打不贏，而老玩家面對高階 AI 卻依舊取得壓倒性勝利，這樣巨大的斷層會帶給格鬥遊戲的初學者巨大的挫敗感。

因此，本專題藉由強化學習方法訓練出能用於格鬥遊戲的 AI，並利用調整獎勵來製作出不同風格、強度的模型，使得各種玩家都能充分享受格鬥遊戲帶來的樂趣。

關鍵詞：格鬥遊戲、深度學習、強化學習

Abstract

In general fighting games with artificial intelligence (AI), there are often problems such as high learning difficulty and serious technical faults between the players. Novices can't even win primary AI, but an old hand still has overwhelming victory against high-end AI, which will bring a huge frustration for the beginners.

In general fighting games with artificial intelligence (AI), there are often problems such as high learning difficulty and serious technical faults between the players. Novices can't even win primary AI, but an old hand still has overwhelming victory against high-end AI, which will bring a huge frustration for the beginners.

Keywords: fighting game, deep learning, reinforcement learning learning

二、問題描述

本次研究要利用強化學習設計適合用在格鬥遊戲之代理人(Agent)，讓 AI 可以透過不斷練習使技巧精進，而在完成此目標的研究過程中會遇到以下兩個主要問題：

1. 如何讓代理人判別出最適合當下狀態使動作，使得整場對戰帶來最大效益，為達成此目標，需要定義好輸入、輸出、模型架構與獎勵函數。

2. 模型將運用在與環境快速交互的格鬥遊戲上，因此，需要讓類神經網路能夠快速地計算出結果。這樣的要求就必須要限制模型的計算量與深度。

三、研究方法

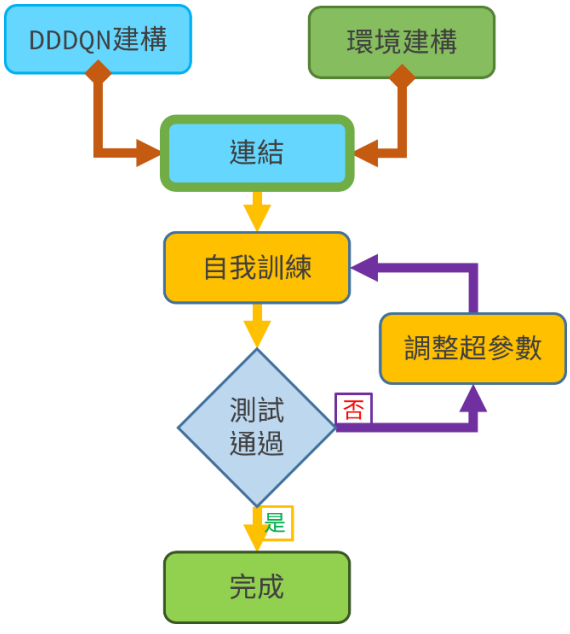

圖 1 開發流程

圖 1 為本專題的開發流程，將使用強化學習算法中的 Dueling Double DQN[3,4,5]作為主要研究方法，並測試各類獎勵函數，使模型能更適合使用在格鬥遊戲上。開發流程的分述如下：

1. 藉由 babylon.js 與 blender 於瀏覽器上建置格鬥遊戲環境。
2. 使用 tensorflow.js[2] 實作 Dueling Double DQN 後與測試環境進行連結。
3. 讓模型自我對戰並開始訓練。
4. 完成訓練後測試模型強度並調整超參數(包含獎勵函數與類神經網路架構)。
5. 重複步驟 3、步驟 4，直到模型表現良好穩定。

四、系統架構

4.1 運作流程

本專題的系統流程如圖 2 所示，當啟動遊戲環境後，網路模型與環境每幀的互動資料就會被收錄進記憶池中，當一回合對戰結束後就會抽取出記憶池內的經驗開始訓練並優化模型。當結束一回的訓練後就會開始新對戰，藉由重複對戰、訓練的循環使網路學習更佳的操作技巧。

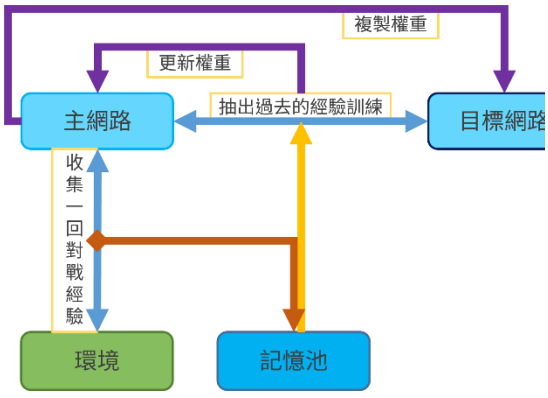

圖 2 系統流程

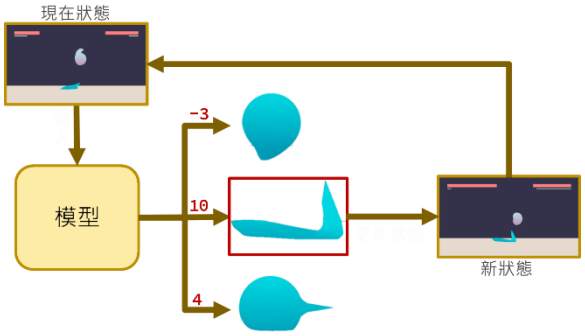

圖 3 決策流程

實際對戰的運行方式如圖 3 所示，當模型估測出所有動作的價值後，就可藉由 Epsilon Greedy 挑選動作，Epsilon Greedy 挑選的作法是有 ε 的機率隨機選擇動作，剩下 1- ε 就挑選有最高價值的動作。在學習階段時可以調高 ε 來增加探索性，讓模型更快的學習到良好的操作方式，而在實際應用時降低 ε 就能使模型穩定發揮所學會之技巧。

4.2 輸入與輸出格式

輸入由[我方狀態向量,我方上一個操作,敵方狀態向量]串接而成，而狀態向量則是由血量、負荷值、位置、跳躍次數、面對方向與目前的動作(與操作不同)等等資訊組合而成。

輸出是<上/下/左/右/輕攻擊/中攻擊/重攻擊>這七個按鍵按與不按的組合所能獲得的分數，共計有$2^7 = 128$個輸出。

五、實驗成果與討論

5.1 實驗環境

表 1 環境配置

處理器	Intel® Core™ i5-4570 CPU @3.20GHz
記憶體	8.00GB
圖形處理器	NVIDIA Geforce GTX 1060 6GB
作業系統	Windows 10 64 位元作業系統、Linux Mint 64 位元作業系統
開發環境	Chrome 77 版以上、Node.js
使用語言	JavaScript
函式庫	Tensorflow.js、Babylon.js

表 1 為本專題的環境配置。運用 JavaScript 跨平台的優勢使本專題能於各作業系統上快速建置開發環境，此外 Tensorflow.js 可利用 webGL API 調用 GPU 的平行運算能力加速計算速，並且不必像 Tensorflow.py 還需要另外安裝 CUDA。

5.2 成果與討論

實驗過程中，如果隨機挑選經驗池中的記憶進行訓練，會導致學習速度緩慢。因此，在實作時增加了簡易的優先經驗重播機制，計算出每筆記憶的預測誤差，讓誤差大經驗的有更高的機率被抽出來變成訓練資料，大大的加快了進步速度。

關於獎勵函數的設定，在經過多次研究後發現，若獎勵函數設計的過於複雜，將會使模型的學習效率低下。為此，將獎勵函數設定為：

if 被敵人擊中：
　負值獎勵(大)
else if 擊中敵人：
　正值獎勵(大)
else：
　我方血量百分比-對方血量百分比

在測試過程中，嘗試將激活函數從 selu[6]換成 mish[7]後，模型的應對能力又更加優異。通過實驗調整過各種超參數的最終版本模型，在歷經 96 小時的自我訓練之後已獲得不錯的應對能力，面對將人類作為對手的場景也不會落入下風，在經過多人測試後得到超過 7 成的勝率。

六、結論

本專題降低了設計格鬥遊戲 AI 的複雜度，只需要調整獎勵函數就可改變 AI 的行動模式，且訓練出來的模型能於對戰中取得不錯的勝率。但此研究最終目的是製造出與玩家一同成長的勁敵 AI，因此，未來將測試不同強化學習方法與各類模型搭配組合後的效果，讓模型學會「放水」，藉由這種方式模擬出「勁敵」，並藉由與勁敵競爭所帶來的刺激感來提高玩家對遊戲的依賴度。

除此之外，本專題目前是將角色的各種狀態作為輸入使用，但這樣會出現以下兩個缺點：

1. 要去額外設計輸入格式，就算經過設計也不一定能正確表達角色的狀態向量。

2. 模型的通用性低下，一個模型只能學習一種對戰組合，如表 2 所示當有 N 個角色就需要N^2個模型。

表 2 角色配對示意

對手 / 我方	角色 A	角色 B	角色 C
角色 A	模型 <A,A>	模型 <A,B>	模型 <A,C>
角色 B	模型 <B,A>	模型 <B,B>	模型 <B,C>
角色 C	模型 <C,A>	模型 <C,B>	模型 <C,C>

因此未來的改進方案希望能結合 word embedding 的方式，讓模型自行學習編碼狀態向量，並如圖 4 所示，將模型拆分成角色狀態編碼、環境映射與角色動作映射三區塊，如此就只需要準備與角色數量相等的<狀態編碼>與<動作映射>，這不僅減少模型的數量還達成更高的通用性。

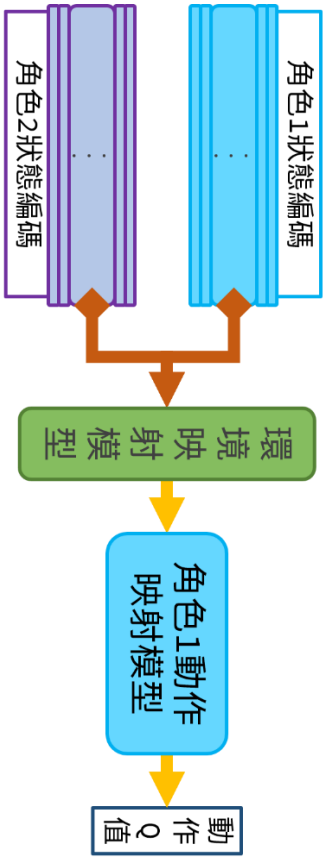

圖 4 未來架構

參考文獻

[1] CS 294 Deep Reinforcement Learning (UC Berkeley), http://rail.eecs.berkeley.edu/deeprlcourse-fa17/, 2017.

[2] Tensorflow.js, https://www.tensorflow.org/js/, 2019.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, “Playing Atari with Deep Reinforcement Learning,” arXiv preprint arXiv:1312.5602, 2013.

[4] Hado van Hasselt, Arthur Guez, David Silver, “Deep Reinforcement Learning with Double Q-learning,” arXiv preprint arXiv:1509.06461, 2015.

[5] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas, “Dueling Network Architectures for Deep Reinforcement Learning,” arXiv preprint arXiv:1511.06581, 2015.

[6] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, Sepp Hochreiter, “Self-Normalizing Neural Networks,” arXiv preprint arXiv: 1706.02515, 2017.

[7] Diganta Misra, “Mish: A Self Regularized Non-Monotonic Neural Activation Function,” arXiv preprint arXiv:1908.08681, 2019.