問題描述

超級馬力歐兄弟是由任天堂於1983年推出的遊戲，截至目前為止，由人類打出來的any%（指不限完成率的情況下速通）成績為4分54.448秒。而Tool-Assisted Speedrun（以下簡稱tas），是指用其他工具，以精度極高的方式完成遊戲，可避免手操上難度要求與運氣因素，其成績為4分54.265秒，是目前人類根據自己的遊戲理解，找出的理論最佳的成績。但我認為20年後的ai，將可能找到tas的更優解。

事實上，早在2018年，ai就已經能自主學習通關這遊戲，然而，嚴格意義上的速通，在目前為止仍遠不如人類玩家。為了打破tas記錄，我認為ai最關鍵的學習方法為強化學習，因為速通的目標為通過一連串的動作序列，以最短時間完成遊戲，ai能人物位置、速度、當前時間等作為資訊來源，從過關時間得到獎勵，不斷試錯找出最短時間。但單純強化學習，ai仍幾乎不可能達到目標，原因如下：

1.行動空間巨大：若單純窮舉，相當於每秒60幀，每幀8個方向與跳躍組合。
2.獎勵延遲：ai在過程中無法知道結果好壞，只有通關後看到時間，才能得到「獎勵」。
3.操作精度要求極大：某些速通技巧，即使操作時機只差一幀，結果也會有極大區別。

下列的模型設計中，只用了強化學習，後續如果想克服上述的問題，我認為能往結合監督式學習方面研究。

模型設計

(代碼有使用chatgpt來幫忙完成)
設計出簡化模型的方式如下：
將遊戲合理的分成大量的起點和終點（如8-4開始區域至下一個水管區域），並分別對每段做出訓練，嘗試短的時間區間內找到最佳動作序列，以到達終點的時間減少作為獎勵。

為模擬AI尋找最短通關策略的核心挑戰，首先建立了一個極為簡化的一維跑關模型(見model.nypb第一部分)。環境共有10個位置（0–9），0為起點、9為終點。格3為坑洞、格5有敵人。每一幀可採取兩種動作：run或jump。若以run落在陷阱或敵人格上則死亡（回報 −100），以 jump 則可安全越過。每步行動消耗 1 點時間（回報 −1），抵達終點給予 +100 獎勵。這是個確定性MDP（Markov Decision Process），可完整定義狀態轉移與報酬函數。

求解方法是使用Value Iteration求得最優策略。其核心方程為$V_{k+1}(s) = \max_a[r(s,a) + \gamma V_k(s')]$，在此$\gamma = 1$，因任務有限步即結束，反覆更新直到所有狀態的價值收斂，接著依每個狀態選取使$r + V(s')$最大的動作作為最優策略。

然而若只用上述簡化模型，極難反映出實際的情況，主要原因是此模型排除了glitch(漏洞)的影響，因此我又設計了另一個模型(見model.nypb第二部分)，以Bullet Bill漏洞現象為例子，讓AI系統探索這類「非預期但可重現」的遊戲漏洞。

在該模型中，AI 以隨機搜尋（Random Explorer）模擬探索過程。探索器會維護一個「archive」用以儲存已發現的狀態，並從中隨機挑選一個儲存點繼續嘗試新動作。若發現新的記憶體狀態，則將其加入 archive。

當環境偵測到warp_byte改變時，即代表觸發了Bullet Bill，系統會將該動作序列記錄為成功範例。這種「回溯式探索」能逐步擴張已知空間，並利用保存點（savestate）重複嘗試新的行為組合，模擬強化學習中的「探索（exploration）」階段。

該模型的核心使用了強化學習，本質是一個馬可夫決策過程，每個動作都會改變環境狀態與回饋獎勵。AI 需在未知環境中透過試錯學習最有效率的策略，最終目標是觸發Bullet Bill（最大化最終獎勵）。這正是強化學習的典型架構。在本模型中，State為位置、子像素、RAM。Action為控制按鍵輸入。Reward為每幀 −1 表示時間成本，觸發Bullet Bill或到達終點則給予高額正獎勵。

在這個模型中，AI 會找到Bullet Bill的唯一情況是：它剛好在一個有特殊物件的frame、角色位置正好重疊該物件、子像素位置 = 0.5、當下動作是「右移＋跳躍」。觸發條件相當嚴苛。

結果

第一個模型順利的找到最優解，在tile2與tile4選擇”jump”決策，其餘使用”run”，AI共用9步（幀）安全抵達終點，總報酬 92 分。

	state	action	next_state	reward
0	0	run	1	-1.0
1	1	run	2	-1.0
2	2	jump	3	-1.0
3	3	run	4	-1.0
4	4	jump	5	-1.0
5	5	run	6	-1.0
6	6	run	7	-1.0
7	7	run	8	-1.0
8	8	run	9	100.0

```
Total reward: 92.0
Steps: 9

Optimal policy per tile: {0: 'run', 1: 'run', 2: 'jump', 3: 'run', 4: 'jump', 5: 'run', 6: 'run', 7: 'run', 8: 'run', 9: None}
```

第二個模型最終模擬的結果為ai於19071次迭代才找到glitch，表示平均要嘗試約20000次才會遇到一次 frame-perfect 的碰撞。每回合最多2000幀，約4千萬幀的嘗試中只有一次成功。換算起來成功率約為0.0000025%。可見在這個簡化模型裡，Bullet Bill屬於極低機率罕見事件。

```
[Explorer] Found bullet bill at iter 19071, steps=145, total_trace_len=182

Archive size: 121
Found bullet bill? True
Readable trace: ['noop', 'noop', 'right', 'right', 'right', 'jump', 'right+jump', 'noop', 'jump', 'jump', 'jump', 'jump', 'jump', 'right', 'noop', 'right
+jump', 'jump', 'noop', 'noop', 'noop', 'noop', 'right+jump', 'jump', 'jump', 'noop', 'right', 'right', 'noop', 'noop', 'jump', 'jump', 'noop', 'jump',
'jump', 'jump', 'noop', 'right', 'jump', 'right', 'noop', 'jump', 'jump', 'noop', 'right+jump', 'noop', 'right', 'right+jump', 'noop', 'right', 'noop',
'jump', 'right', 'jump', 'jump', 'jump', 'jump', 'jump', 'right', 'noop', 'noop', 'noop', 'noop', 'right', 'noop', 'noop', 'right', 'jump', 'right+jump',
'right', 'noop', 'noop', 'right+jump', 'noop', 'noop', 'right', 'jump', 'right', 'right', 'right+jump', 'jump', 'right+jump', 'noop', 'jump', 'noop', 'ju
mp', 'jump', 'jump', 'right+jump', 'noop', 'noop', 'noop', 'jump', 'jump', 'right', 'right', 'noop', 'right', 'noop', 'right', 'right+jump', 'noop', 'jum
p', 'jump', 'jump', 'noop', 'right', 'noop', 'jump', 'noop', 'jump', 'jump', 'jump', 'noop', 'jump', 'noop', 'jump', 'right', 'right', 'right', 'jump',
'right+jump', 'noop', 'jump', 'right', 'jump', 'noop', 'jump', 'noop', 'jump', 'noop', 'jump', 'right', 'jump', 'right', 'jump', 'noop', 'jump', 'jump',
'noop', 'right+jump', 'noop', 'jump', 'noop', 'right', 'jump', 'jump', 'right', 'right', 'right+jump', 'right', 'jump', 'right+jump', 'right+jump', 'noo
p', 'jump', 'noop', 'right+jump', 'jump', 'right', 'noop', 'noop', 'jump', 'jump', 'right', 'noop', 'right+jump', 'right', 'jump', 'noop', 'right+jump',
'noop', 'noop', 'noop', 'noop', 'jump', 'jump', 'jump', 'jump', 'right+jump', 'jump', 'right', 'right+jump']
Final obs: {'pos': 14, 'subpixel': 0.5, 'ram': (8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), 'frame': 182, 'special_positions': (14,)}
Info: {'warp_byte': 8, 'collided_special': True}
```

討論

此模擬模型中，雖然ai成功觸發了bullet bill，但是在中途經歷了大量的行動，才偶然的找到glitch，展示了AI難以從稀疏、延遲的回饋中學到最優解。此外，使用上述方法訓練出的AI只知道「哪裡」出錯，而不知道「為什麼」出錯，導致即使找出glitch，也很再做後續找出更快的路徑。

不過，這樣的模型仍然有意義，說明利用ai發現新的未知glitch完全是有可能的事情，而這是突破紀錄最有可能的方式，後續可人為研究並重現glitch，再導入監督式學習內容優化路徑，我認為未來的ai真能應用在tas上。

若未來ai真突破tas記錄，其影響力遠不止在遊戲方面。像是能說明ai可在極小誤差容忍下，找到全域最優解，這在組合爆炸問題、程式最佳化等方面也能取得重大突破。同時說明ai具備改進演算法的能力，不僅能夠學習，還能發明新策略，甚至可能在創造性上超越人類。