清華電機實作專題成果展 網頁成果摘要

Website Abstract for Undergraduate Project Exhibition

A Deep-Reinforcement-Learning Based High Frequency Trading Strategy
基於深度強化學習之高頻交易策略

專題領域： 系統領域
組　　別： A390
指導教授：馬席彬 教授
組員姓名：周光祈、歐羿辰、黃霈紳

1. 報告摘要：

隨著電腦技術和人工智慧的發展，高頻交易的穩定性和效益受到越來越多的關注。過去的文獻中，已有許多深度強化學習在這個領域的應用。在本研究中，我們探索了深度強化學習在這個領域的潛力，並試圖利用相關的演算法來開發高頻交易策略。 我們提出了一種基於近端策略優化演算法實現的交易代理人策略，這是一種由兩個神經網路組成的演員評論家演算法。它的輸入是逐筆交易資訊，輸出是針對目前資訊的交易行為。我們的目標是在考慮交易成本的前提下，對策略評估指標有良好的表現。此外，我們的策略中還利用了數據儲存結構來幫助更新輸入狀態，這是本研究的重要設計之一。最後，我們對策略模型的運算效率進行了改善，將其轉換為量化的定點數運算，並對不同的量化位元數進行了效能測試。實驗結果表明，我們提出的方法在模擬交易環境中取得了正面的效益。

2. 報告內容：

(1) 研究動機 & 目的

隨著深度學習(Deep Learning)在近年的崛起，其在金融領域的應用得到卓越的成效。其優勢不僅在於發掘數據中的相關性，並且在高維度的數據(High Dimensional Data)分析有強大的潛力，在過去的研究中，能夠成功的預測市場未來價格趨勢，而有助於策略見微知著，及早做出反應而取得有利的地位。

強化學習(Reinforcement Learning, RL)作為機器學習的分支，其框架對於Decision-Making Problem 類型的任務非常有效，透過代理者(Agent)和環境(Environment)反覆進行互動，以取得最大獎勵(Reward)為目標而學習最佳的行動策略(Policy)，正好對於高頻交易這類缺乏正確的行動規則的任務展現良好的適性。深度學習與強化學習的結合: 深度強化學習(Deep Reinforcement Learning, DRL)結合雙方的優點，藉由深度學習強大的發掘高維度數據的相關性能力，幫助強化學習演算法直接從這些數據學習其行動策略。因此，本研究選擇採取深度強化學習的訓練方式，搭配些許資料結構來設計交易策略。

然而，在軟體平台上開發的策略通常無法滿足實務的速度要求，將軟體架構實作到硬體上加速是常見的策略驗證手段。因此，本研究所提出的模型有另一個目標，在於實現於硬體上，其必要之軛在於脫離浮點數運算，為了利用較有效率的整數運算，本研究參考文獻設計了模型參數量化的方法，並且探討量化位元數對於模型效能的影響。

(2) 研究方法 Method

我們先對資料與環境做預處理，並設計合理的 Reward 對 PPO 進行訓練，並最後量化整個 Network。

a. 資料與預處理：

本研究所使用的交易數據來自台灣期貨交易所的逐筆行情交易資訊，交易所的委託簿中任何新增、刪除或是修改，交易所都會根據事件發生順序傳送委託簿揭示訊息，本研究使用到的委託簿資料包含了限價委託簿的更新時間、買方(Bid Side)的最佳五檔價格與數量，以及賣方(Ask Side)的最佳五檔價格與數量。本次研究的實驗數據來自實驗室留存的資料，數量僅有六天，分別是2022年的6月13、14日以及10月24~26日、31日。

本實驗選用的期貨商品為小型臺指。小型臺指最小變動為1點，每點價值新台幣50元，也就是每口小型臺指漲跌幅1點即會對投資者造成50元的損益。根據期交所公布的交易費率表，小型臺指的交易成本大概可分為交易經手費每口7.5元、結算手續費每口5元、期貨交易稅如 Eq. 1，P_{tax}為期貨交易稅，P_A為契約價值。一般投資者透過期貨商進行下單仍需要負擔額外的手續費，但這些手續費計算方式依期貨商而有所不同，因此我們以期貨商的角度出發，僅計算必要的交易經手費、結算手續費以及期貨交易稅。

$$P_{tax} = P_A * 0.00002, \qquad (Eq.1)$$

因易受到國外事件影響，且這些事件多發生在台灣市場非交易時段，故開盤後的一段時間市場的價格受到影響最為劇烈。另外，在接近收盤的時候，執行當沖交易的投資者需要在收盤前平倉，因此委託簿的變動相較其他時段更為劇烈。基於上述原因，這兩段時間的交易數據和委託簿的價量相關性較小，因此較不適合做為模型的訓練資料，我們決定只保留開盤以後45分鐘到收盤以前45分鐘的委託簿

資訊做為訓練資料，也就是每個交易日保留上午9:30到下午1:00這段期間的資料，其餘時段捨棄不用。

Data normalization 有助於加速模型收斂。由於交易資料是個序列型數據，我們不能遍歷完整的序列以後取其平均值和標準差做 Normalization，這樣做會使得要做為 Testing data 的資料喪失其未知性。因此對於每個交易日，我們取其開盤價做為當日的價格中心值，至於價格偏差值，我們統計所有交易日數據，選取距離常見的偏差值最接近的2指數，做為所有交易日的偏差值，在本研究中，偏差值 $P_{dev} = 2^6 = 64$。Normalization 如 Eq. 2，其中P_{open}為經過 Data Scaling 的開盤價。

$$P_{normalized} = \frac{P_{scaled} - P_{open}}{P_{dev}}, \qquad (Eq. 2)$$

b. 近端策略優化演算法(PPO)：

假設在一個情節中代理人接收了N個狀態，並做出N個動作。其中，第t個狀態、動作以及對應的獎勵分別為s_t、a_t 、r_t ，那麼第t時間的累積獎勵G_t計算方法如 Eq. 3。其中，γ 是折扣因子(Discount Factor)，$0 < \gamma < 1$。

$$G_t = \sum_{n=t}^{N} \gamma^{n-t} r_n, \qquad (Eq. 3)$$

PPO 使用到演員評論家(Actor-Critic)方法，評論家與演員都是一個神經網路。評論家會根據演員接收到的狀態還有目前演員內部的參數，來預測演員在當前狀態能夠獲得多少累積獎勵。我們會把演員執行動作後得到的累積獎勵減去評論家的預測值當作優勢函數(Advantage Function)。優勢函數的值越高表示該動作越好，我們就可以調整演員參數，讓執行此動作的機率提高。

近端策略優化演算法使用異策方法(off-policy)，異策指用來與環境互動的模型與被訓練的模型不是同一個，並且同一筆與環境互動蒐集的資料可以拿來訓練模型數次，但仍有限制：兩個模型的行為機率分佈差異必須在一定範圍內，否則會導致訓練成效不好。其想要最大化的目標函數如 Eq. 4。

$$L_t^{CLIP}(\theta) = \min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t),\ clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)},\ 1-\varepsilon,\ 1+\varepsilon\right) A^{\theta'}(s_t, a_t)\right), \text{(Eq. 4)}$$

其中θ'是與環境互動的模型，θ 是要被訓練的模型， $p_\theta(a_t|s_t)$表示模型在接收到狀態s_t後，輸出動作a_t的機率，$A^{\theta'}(s_t, a_t)$表示給定狀態s_t，輸出動作a_t的優勢函數，ε稱作 Clip parameter。可以看到如果當前的優勢函在數是正數，代表此動作是好的，我們就會想增加它被執行的機率。相反地，如果當前的優勢函數是負的，表示此動作不好，我們就會想要降低它發生的機率。此公式中，我們會計算兩項目標函數，然後選擇較小的那一項。第二項目標函數中，修剪函數會把輸出

的值侷限在$(1-\varepsilon)$與$(1+\varepsilon)$之間。當優勢函數大於零時，我們會期望增加$p_\theta(a_t|s_t)$，但我們限制其不能超過$p_{\theta'}(a_t|s_t)$的$(1+\varepsilon)$倍，避免兩模型之間的差異過大。相對應地，當優勢函數小於零時，我們希望降低$p_\theta(a_t|s_t)$，但不能低於$p_{\theta'}(a_t|s_t)$的$(1-\varepsilon)$倍，同樣避免兩模型之間的差異過大。在實作中，實際要最大化的最終目標函數如 Eq 5。

$$L^{CLIP+VF+S}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)], \qquad \text{(Eq 5)}$$

$$L_t^{VF}(\theta) = \left(V_\theta(s_t) - V_{\theta'}(s_t)\right)^2, \qquad \text{(Eq 6)}$$

其中c_1和c_2是兩個正數，c_2可稱作 Entropy coefficient，是可以調整的超參數。可以看到最終目標函數由三個部分組成，第一部分為先前介紹經過修剪之後的目標函數，第二部分為評論家對於兩模型看到狀態s_t之後，評估兩模型可以獲得累積獎勵之平方差。當此平方差越大，表示兩模型的差異越大，所以要最小化此項。第三項為被訓練模型接收到狀態s_t之後，動作空間的亂度。我們希望避免模型只執行單一動作，而沒有嘗試過其它動作是好是壞。模型擁有越多亂度代表模型越有機會嘗試不同動作組合，進而找出最好的策略，所以要最大化此項。

c. 交易環境、Action、Reward

由於我們只能模擬交易環境，因此假設模型的交易行為並不會對市場產生影響的前提下，為了貼近真實的交易環境，避免模型行為出現滑價情形而導致預期外的損失，我們將模型每次行動的最大交易量定為一口，並且使用市價單交易，即使用最佳買賣價成交。模型的 Action 列表如 Table 1:

Table 1 Action Space of Model

Action Number	Description
0	不做任何事情
1 (Buy)	以當前的P_{Ask}買入一口
2 (Sell)	以當前的P_{Bid}賣出一口

獎勵函數(Reward Function)將針對模型的行動(a_t)做出評價，而評價交易行動的好壞通常使用該行動所產生的損益(Profit and Loss, PnL)來進行評估，計算損益就需要一段時間的價格變化，理想上應該要計算當前的行動在未來能產生的損益，是最符合投資者的思維，但是由於我們對於未來的價格並不能精準預測，只能採用估計的方式，我們認為這樣計算的獎勵數字，對於模型來說隱藏太多資訊導致學習效果有限。不過，我們可以精準得到過去模型遇見過的價格，因此可以利用過去和當前的價差來計算當前獎勵(r_t)，如 Eq. 7所示:

$$r_t = \begin{cases} 0, & if\ a_t = 0 \\ \mathrm{A} * \Delta P_{buy} - \alpha, & if\ a_t = 1, \\ \mathrm{A} * \Delta P_{sell} - \alpha, & if\ a_t = 2 \end{cases} \quad (Eq.7)$$

其中α是該行動a_t的預期成本，$A = \frac{C}{\log_2(2+|Q_t|)}$做為損益對於成本的放大倍率($C$是常數，$Q_t$是當前持倉量)，因為高持倉量需要花更多時間平倉是不理想的，我們想要降低高持倉量時行動得到的好處。$\Delta P_{buy} = P_{Bid,past} - P_{Ask,t}$，若過去曾經用$P_{Bid,past}$賣出一口，而在當下時間為$t$時買進一口將得到$\Delta P_{buy}$的利益，$\Delta P_{sell} = P_{Bid,t} - P_{Ask,past}$，若過去曾經用$P_{Ask,past}$買進一口，而在當下時間為$t$時賣出去一口將得到$\Delta P_{sell}$的利益。

而過去的買賣價$P_{Bid,past}$和$P_{Ask,past}$的取得方式，我們使用 Priority Queue 儲存過去的價格，如 Fig. 1。每當有一個 Buy 或 Sell 的行動時都會更新兩個 Priority Queue，而這兩個 Queue 所儲存的都是模型曾經做過交易的價格。如果行動是 Buy 並且 Priority Queue (Sell)不是空的，那麼就 Delete 其中 Top 元素，也就是所儲存的最高 Bid Price；而如果 Priority Queue (Sell)是空的，就 Insert 這次的 Ask Price 到 Priority Queue (Buy)中。如果行動是 Sell 並且 Priority Queue (Buy)不是空的，那麼就 Delete 其中 Top 元素，也就是所儲存的最低 Ask Price；而如果 Priority Queue (Buy)是空的，就 Insert 這次的 Bid Price 到 Priority Queue (Sell)中。$P_{Bid,past}$即為 Priority Queue (Sell)的 Top 元素，代表所持倉的最高 Bid Price。$P_{Ask,past}$即為 Priority Queue (Buy)的 Top 元素，代表所持倉的最低 Ask Price。

以上述方式維護這兩個 Priority Queue，同時可以讓其中所儲存的元素數目和持倉量保持一致，因此若兩個 Queue 都是空的，代表模型的行動結清所有過去的買賣，完成平倉，此時就令$P_{Ask,past} = P_{Bid,t}, P_{Bid,past} = P_{Ask,t}$，使得後續的$\Delta P_{buy}$和$\Delta P_{sell}$代表的是時間$t'$和時間$t$的價差$(t' > t)$，$\Delta P_{buy} > 0$代表價格下跌適合買入；$\Delta P_{sell} > 0$代表價格上漲適合賣出。有了 Action Space 和 Reward Function，剩下 State (Observation Space)尚未定義，我們主要提供價格的資訊，同時提供價差如 Bid-Ask Spread 和ΔP_{buy}、ΔP_{sell}等等，另外提供持倉量的資訊給模型，詳細 State 如 Eq. 8，其中Q_M是最大持倉量:

$$s_t = \begin{bmatrix} (P_{Bid,t}, P_{Ask,t}) \\ P_{Ask,t} - P_{Bid,t} \\ (\Delta P_{buy}, \Delta P_{sell}) \\ P_{Bid,past} - P_{Bid,t} \\ P_{Ask,t} - P_{Ask,past} \\ Q_t / Q_M \end{bmatrix}, \qquad (Eq.8)$$

d. 神經網路架構

在 Actor-Critic 神經網路設計，採用三層的 Fully-Connected (FC) Layer 架構，並且 Hidden Layer 的 Activation Function 採用 Leaky ReLU 設計，一方面 ReLU 函數在硬體實作上十分簡易，另一方面變形的 Leaky ReLU 處理負數輸入的時候不會完全關閉神經元而產生“Dying ReLU”的問題，對於模型輸入 State 中有正有負的

情況較 ReLU 合適，示意圖如 Fig. 2。

e. Action-Reversing

由於獎勵函數制定建立在一個前提，就是在a_t之前模型曾經有行動過，累積一定的反向持倉量，否則如果遇到價格持續大漲或是持續大跌的情形，模型很有可能會快速累積持倉，並且沒有機會平倉，導致大量的買高賣低情況發生而產生虧損。為了應付這種情況，我們需要在交易前期製造反向持倉，我們發現如果當日的價格是大漲的，那麼當日的收盤價通常會高於開盤價，而如果是大跌的話，收盤價會低於開盤價。照著這個邏輯，我們在開盤價上下定一小段範圍(這是個要預調的參數)，當價格落在這個範圍中時，將模型的買入行動改成賣出，賣出行動改成買入，就有機會製造反向的持倉，直到價格漲破或是跌破這個範圍，模型的正常行動就可以將這些反向持倉進行平倉。我們將這個方法稱做“Action-Reversing”。

f. Post Training Quantization (PTQ)

Post Training Quantization(PTQ) 的主要優勢在於無需重新訓練模型。它允許對現有模型進行直接的量化優化，節省時間和計算成本。同時，因為在我們用 floating points 訓練 Agent Network 時發現對於設計的模擬環境，Network 參數極其敏感，也因為用了不同天數(資料變化可能很大)的資料 Fine-Tune 訓練出來的，且訓練本身花費時間極長，故最後選擇用已經訓練好的模型做 PTQ。

首先因為輸入層的分布有正有負而正負的幅度沒有差很多，故我們選擇採用 Symmetric Mode (Zero point $Z = 0$) 之量化方法: $q = round\left(\frac{r}{S}\right),\ r = Sq,\ S = \frac{r_{max} - r_{min}}{2^B - 1}$ 的 quantization scheme，其中 r為 de-quantized value (floating point)，q為 quantized value (fixed point)，S為 scaling factor，B為 bit 數。其次，考量到 input 及各層 weight, bias 的特性，我選擇使用 $r_{max} = mean(r) + 3std(r),\ r_{min} = mean(r) - 3std(r)$，並將離群值 saturate 到$r_{min}/r_{max}$

Actor Network 的 Fully-Connected Layer (FC)中的矩陣乘法 quantized 後可表示成 Eq. 9。

$$q_{out}^{(i,k)} = \frac{S_W S_I}{S_O} \sum_j q_W^{(i,j)} \times q_I^{(j,k)}, \qquad (Eq.9)$$

其中 i, j, k代表矩陣的行列元素，S_W為該層的 weight scaling factor，S_O 為該層的 output scaling factor， S_I 為該層的 input scaling factor，注意到任一層的 S_I 即為其上一層的S_O。

(3) 研究結果 Results

對於交易策略有非常多種評估指標可以判斷策略的好壞，在本次實驗中，我們選擇兩種重要的指標來評估所提出的交易策略。

- 每日稅後淨利率(Daily Net Profit Margin):

  扣除交易成本以後的淨利直接反映了投資策略的獲利成效，也代表策略的好壞，其計算方式如Eq. 10，P_{net}為淨利率，A_{settle}為當日收盤後結算的總資產，$A_{initial}$為當日初始的總資產。

$$P_{net} = \frac{A_{settle} - A_{initial}}{A_{initial}}, \qquad (Eq. 10)$$

- 勝率(Win Ratio):

  勝率反映了策略獲利次數的比例，對於高頻交易多次進出場的交易策略，獲利的次數也變得相當重要。不過勝率也會隨著交易風格不同而變化，如果策略將商品閒置以待利益累積到一定程度才進行平倉，追求單次進出場的獲利上限，則對勝率的要求不會太高。勝率計算如Eq. 11，W代表勝率，T_w代表回測期間獲利的進出場次數，而T_t代表回測期間的總進出場次數。

$$W = \frac{T_w}{T_t}, \qquad (Eq. 11)$$

  Fig. 3, 4展示了我們的成果。可以看到 Action-Reversing 關鍵的影響在 Win Ratio 的變化，Win Ratio 沒有變化的交易日代表其價格範圍在時間9:30以後都超過 Action-Reversing 起作用的範圍。只要 Win Ratio 有變低，當日的 Daily Net Profit Margin 就會大幅提升(10/24/2022例外，但其損失相比其他交易日獲得好處小很多)，事實上這幾日的價格都有出現大幅度的漲跌，也證實 Action-Reversing 可以在這種情況將模型的行動修正。

  除此之外，可以看到在沒有 Action-Reversing 的情況下，雖然多數交易日利潤不理想，但是模型的行動 Win Ratio 非常高，並且在某些價格震盪的交易日可以獲得利潤(如6/13/2022、10/24/2022)，代表模型適合的市場情況是小幅度並且快速的價格漲跌，使得模型頻繁進出場獲取微小的利潤。因此如果能夠事先預測當日的市場情況，而決定是否要對模型施加 Action-Reversing，將能夠各取所長。本實驗尚未對此有所鑽研，但是這是未來優化模型的可行方向。

  將各層 Network quantized 後，並用位元數對於結果作圖，找出幾乎不影響效能下的最低 bit 數如 Fig. 5，從圖中可以看出約是8~10 bit 之間。

  可以看到從原本每層都是 full precision(32-bit) 到最後只用了$(8bit, 5bit, 8bit)$，節省相當多運算量，並且就測試結果來說效能幾乎沒有影響。而如果我們放寬限制讓他只要保留了80%以上的效能(針對 Daily Net Profit Margin) ，可以進一步縮小到$(8bit, 3bit, 8bit)$，其效能比較呈現如 Table 2。其中 Result 2是允許20%效能損失的結果。

  Table 2 Comparison of Performance Between Full Precision and Quantized ones

	Full precision			Result 1			Result 2		
Bit width	FC1	FC 2	FC 3	FC1	FC 2	FC3	FC1	FC 2	FC 3
	32	32	32	8	5	8	8	3	8
Win rate	82.93%			82.86%			86.05%		
Net Profit	10.59%			9.86%			8.06%		

(4) 總結 Conclusion

本次專題結合深度強化學習搭配 Priority Queue 資料結構，利用到過去的交易訊息，輔以條件判斷方法，應用在逐筆交易資料有初步的成效，PPO agent 能夠成功學習到最大化所設計獎勵函數之行動策略。實驗結果也呈現出此模型的潛力。而後續模型的量化，我們也能夠尋找到最佳量化位元寬度，大幅提升運算效率和縮小模型大小，而幾乎不影響原模型的效能。

然而，在這次專題仍未考量到關於市場的許多變化，例如限價委託簿和成交訊息等有用的資訊，面對真實的交易環境還存在許多挑戰。此外，如何將這次專題所運用到的架構實際做到硬體加速，過往文獻雖然有對於這方面的研究，但都仍需要更深入的研究。

3. 參考圖片：

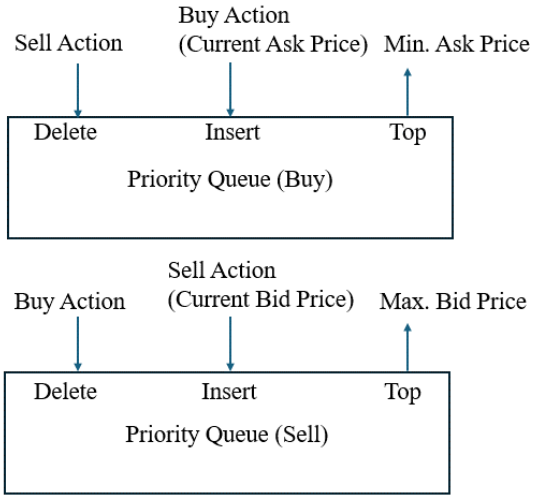

Fig. 1 Priority Queues used for Storing Past Price

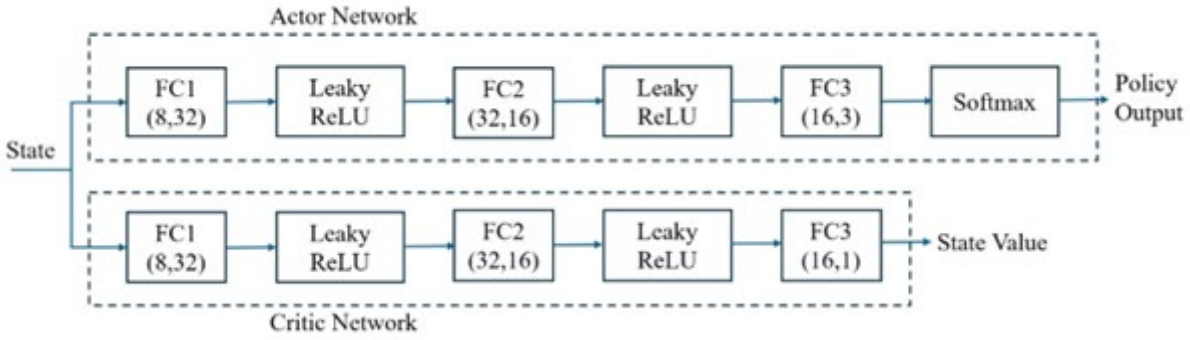

Fig. 2 Framework of Actor-Critic Neural Network.

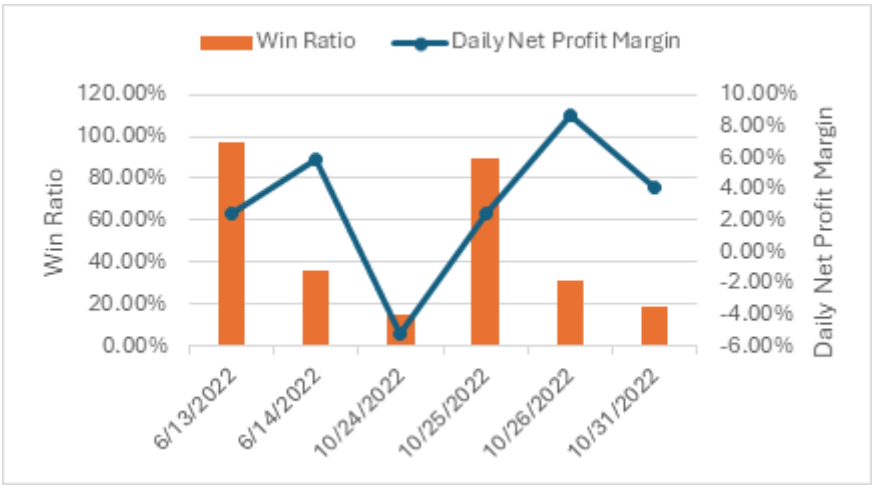

Fig. 3 Performance on 6 Trading Days.

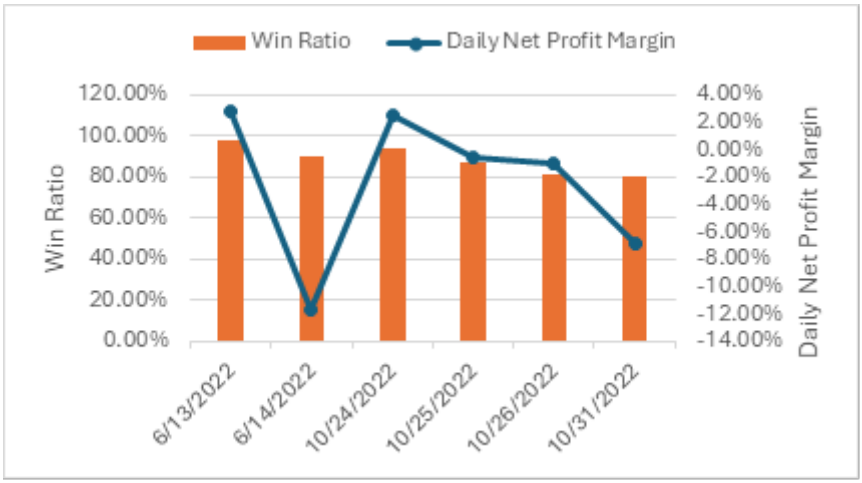

Fig. 4 Performance on 6 Trading Days Without Action-Reversing.

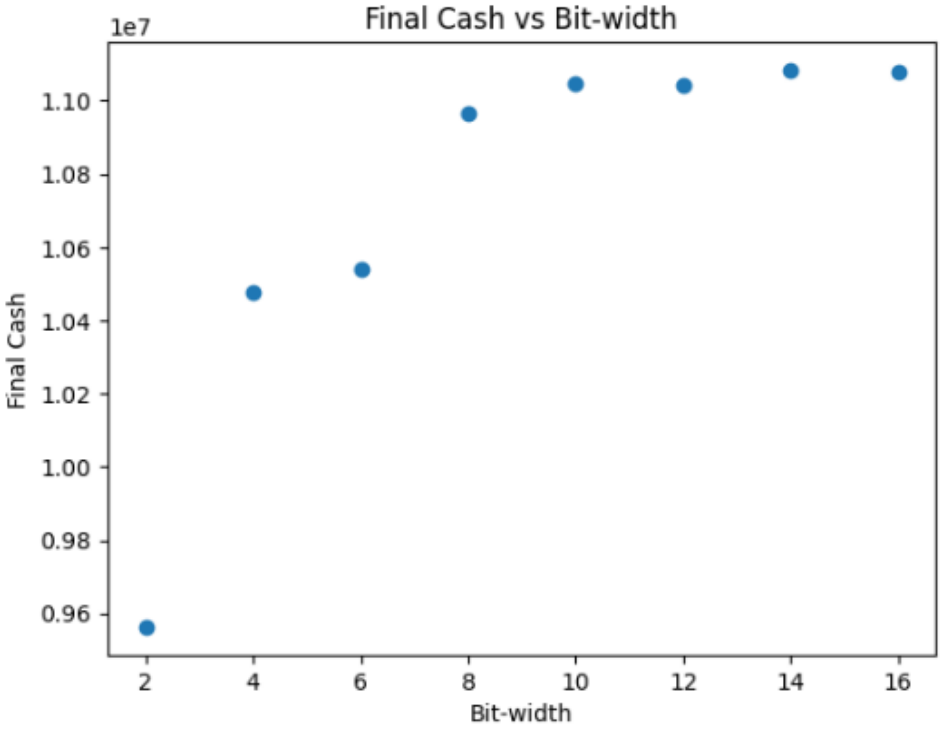

Fig. 5 Performance Corresponded with Arbitrary Quantization bits.

4. 心得感想：

最初選擇這個專題時，只是覺得設計個策略應該可以順利完成，然後可以實作硬體加速。不過真正開始研究後，才發現從環境的建立到建構模型，其中都藏著種種阻礙，並且設計策略時一開始成效非常不好。很高興馬席彬教授每周對於我們進度的掌握，並給出適時的建議，讓我們對於如何調整我們的設計有方向可以依循。研究的過程，我們組員實作能力不一，有的不熟悉程式語法，過程中都需要摸索，並且強化學習相較一般監督式學習更為複雜，訓練的過程非常陌生，很高興我們組員間達成配合，才能成功完成強化學習框架。也很高興當初選擇這個題目，雖說最後沒有實際實作硬體加速，不過能透過策略設計學習到的研究和實作技巧，也相當有收穫，我們深信專題的成果對於未來能有所幫助。最後，要感謝馬席彬教授的指導，經過這幾個月，我們收到不少有幫助的建議，雖然說研究是我們獨立完成，但少了他的幫助的話，我們肯定會繞不少遠路。