A low-latency recurrent neural network that forecasts the liquidity of TAIEX Futures

用低延遲的循環神經網路預測臺股期貨流動性

組別：A226 指導教授：馬席彬教授 組員：林靖 周星航 林致寬

Abstract

近幾年金融的交易許多都建立在軟體的運算，為了搶得先機大家不斷的改進電腦設備，獲得更好的運算速度。整個交易系統的流程是先透過網路取得封包並解讀，根據解讀出來的資訊維護系統內的行情資料，然後再根據交易策略決定行動。因為行情封包的格式、交易的策略等都是固定的所以可以透過把系統建立在 FPGA 上以獲得更好的反應時間。

本實作專題探討的自動交易系統有兩項特色：深度學習和硬體加速。期貨操盤手根據交易經驗所設計的交易策略不易量化或最佳化，而深度學習可以解決人為設計的盲點。用深度學習模型作為交易策略可以勝過市場上的其他交易競爭者，提高收益。市面上以軟體實作的自動交易系統已經成熟穩定，但是軟體的缺點是延遲較高，無法在詭譎多變的期貨市場中搶得先機。因為硬體加速平台不像軟體有中央處理器和作業系統等速度瓶頸，所以可以較迅速的應對市場突發事件。

我們訓練出預測買賣價差的 LSTM 模型，準確度為56%。使用高階合成方法進行實作，我們的硬體可以在大約40奈秒內完成單次預測之運算。

Introduction

解封包取得交易資料

OSI 模型是網路的標準框架，包含實體層、資料連結層、網路層、傳輸層等。因為市面上量產的網路介面控制器(NIC)支援各種實體層和資料連結層的通訊協定，所以在設計時並沒有針對臺灣期貨交易所的封包最佳化。而作業系統的協定疊(protocol stack)則含有更多不利於高頻交易的因素。除了中央處理器造成軟體的速度瓶頸之外，還會因為與網路不相關的其他軟硬體，導致中斷

(interrupt)和快取未中(cache miss)，使得延遲難以評估或時好時壞。

逐筆行情資訊傳輸作業手冊定義了完整的行情封包格式，因為 Ethernet II、IPv4、TCP 等通訊協定的 header 有記載 payload 的長度，所以可以只讀取封包中特定位置的位元，就計算出 payload 的第幾個位元是價量資訊。收到一個行情封包之後，不需要完整解出所有 OSI 模型中定義的欄位，也可以取得特定商品的價量資訊。

長短期記憶單元的輸出和輸入訊號

長短期記憶是一種循環神經網路。長短期記憶的每一層根據時間為 t 時的隱藏狀態 h 和輸入 x，計算時間為 t+1 時的單元狀態c_{next}和隱藏狀態h_{next}。為了描述長短期記憶的每一層計算過程，定義：

$W_i \in R^{h\times d}$ 、 $U_i \in R^{h\times h}$ 、 $b_i \in R^h$ ： 訓練出來的 i 閘參數
$W_f \in R^{h\times d}$ 、 $U_f \in R^{h\times h}$ 、 $b_f \in R^h$ ： 訓練出來的 f 閘參數
$W_o \in R^{h\times d}$ 、 $U_o \in R^{h\times h}$ 、 $b_o \in R^h$ ： 訓練出來的 o 閘參數
$W_g \in R^{h\times d}$ 、 $U_g \in R^{h\times h}$ 、 $b_g \in R^h$ ： 訓練出來的 g 閘參數

$i \in (0,1)^h$ ： 時間為 t 時的 i 閘
$f \in (0,1)^h$ ： 時間為 t 時的 f 閘
$o \in (0,1)^h$ ： 時間為 t 時的 o 閘
$g \in (-1,1)^h$ ： 時間為 t 時的 g 閘

$x \in R^d$ ： 時間為 t 時的輸入
$c \in R^h$ ： 時間為 t 時的單元狀態
$h \in (-1,1)^h$ ： 時間為 t 時的隱藏狀態
$c_{next} \in R^h$ ： 時間為 t+1 時的單元狀態
$h_{next} \in (-1,1)^h$ ： 時間為 t+1 時的隱藏狀態

長短期記憶的每一層計算可以分成兩個過程來描述。首先，長短期記憶的每一層根據時間為 t 時的隱藏狀態 h 和輸入 x，計算時間為 t 時的 i 閘、f 閘、o 閘、g 閘：

$$i = expit(W_i \times x + U_i \times h + b_i)$$
$$f = expit(W_f \times x + U_f \times h + b_f)$$
$$o = expit(W_o \times x + U_o \times h + b_o)$$
$$g = \tanh(W_g \times x + U_g \times h + b_g)$$

其中 × 是 matrix multiplication。然後，長短期記憶的每一層根據時間為 t 時的

單元狀態 c、i 閘、f 閘、o 閘、g 閘，計算時間為 t+1 時的單元狀態c_{next}和隱藏狀態h_{next}：

$$c_{next} = f * c + i * g$$
$$h_{next} = o * \tanh(c_{next})$$

其中 $*$ 是 element-wise product。

Limit Orderbook and Order Flow

Orderbook: 在期貨交易市場中，Orderbook 包含一項商品的所有委託單的買入價、買入口數、賣出價與賣出口數，並用檔數表達該筆委託的買入價或賣出價是第幾佳，而 limit orderbook 則只包含該商品前幾檔買入價與賣出價和對應的口數。

Order Flow: 定義$bOF_{t,i}, aOF_{t,i}$分別為第 i 檔買價和賣價在時間 t 的 order flow，我們可透過下列公式獲得各時間 t 的bOF_t和aOF_t

$$bOF_{t,i} := \begin{cases} v_t^{i,b}, & if\ b_t^i > b_{t-1}^i \\ v_t^{i,b} - v_{t-1}^{i,b}, & if\ b_t^i = b_{t-1}^i \\ -v_t^{i,b}, & if\ b_t^i < b_{t-1}^i \end{cases}$$

$$aOF_{t,i} := \begin{cases} v_t^{i,a}, & if\ a_t^i > a_{t-1}^i \\ v_t^{i,a} - v_{t-1}^{i,a}, & if\ a_t^i = a_{t-1}^i \\ -v_t^{i,a}, & if\ a_t^i < a_{t-1}^i \end{cases}$$

其中$b_t^i, v_t^{i,b} a_t^i, v_t^{i,a}$,分別為在時間 t 第 i 檔的買入價和口數以及賣出價和口數。系

系統的特性與輸出和輸入

預測期貨走勢的模組包含單層 LSTM 和單層 MLP。其中 LSTM 單元有單元狀態 c 和隱藏狀態 h 等內部狀態，我們可以選擇使用靜態變數來記憶這些內部狀態，或是將內部狀態外部化。高階合成支援靜態變數，且預設使用存取速度受限的記憶體來合成這些靜態變數，所以使用靜態變數是可行的做法。如果想要將內部狀態外部化的話，則可以替模組新增一組輸入和輸出，改由模組外部的記憶體管理狀態資訊。

我們最後選擇將這些內部狀態外部化，這樣在使用資料中心加速器卡驗證此模組時，可以隨時確認已外部化的內部狀態。如果需要回朔到特定狀態的話，也可以隨時竄改當下的記憶體內容，以提供待測模組指定的狀態資訊。將內部狀態外部化之後，預測期貨走勢的模組就沒有內部狀態了。此模組的輸出完全由當下的輸入決定，與前幾次的輸入無關，所以屬於無記憶性系統 (memoryless system)。

預測期貨走勢的模組有三個輸出(output、cell_state_next、hidden_state_next)和三個輸入(input、cell_state、hidden_state)。其中有四個與狀態有關的訊號，這

四個狀態訊號和 LSTM 的數學模型相互對應。cell_state 對應c，cell_state_next 對應c_{next}，hidden_state 對應h，hidden_state_next 對應h_{next}。而模組的輸入訊號 input 是 20 個 16 位元定點數，這 20 個定點數是買賣雙方的五檔價量。模組的輸出訊號 output 則是 3 個 16 位元定點數，表示預測的走勢。

在 LSTM 數學模型中，c_{next}會存回c，h_{next}會存回h。而我們實做出來的模組沒有內部狀態，所以需要額外連接足夠數量的暫存器來儲存 cell_state_next 和 hidden_state_next，並在下一次委託簿更新時，隨同更新後的委託簿一起送到模組的輸入端。

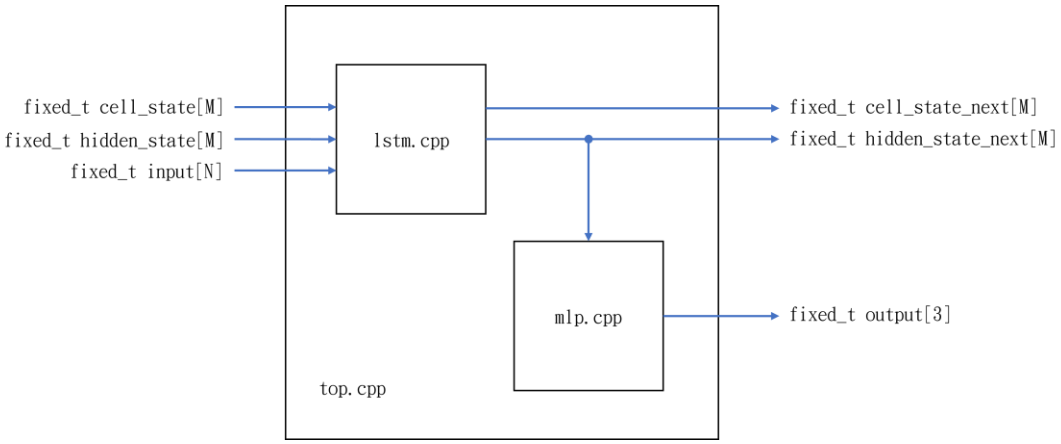

圖 1：TOP 模組的方塊圖。資料來源：專題組員自行繪製。

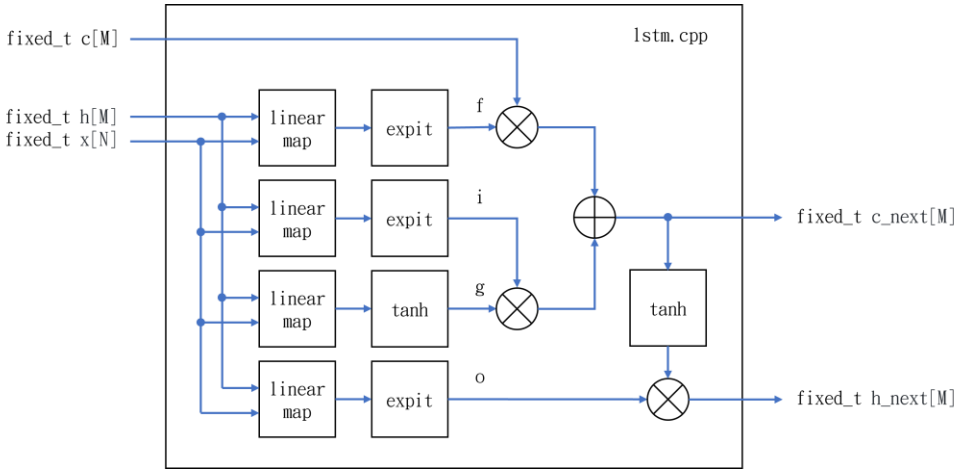

圖 2：LSTM 模組的方塊圖。資料來源：專題組員自行繪製。

硬體加速設計

在 LSTM 模型中大量運用到 *tanh* 、 *sigmoid* 和矩陣乘法，所以我們特別針對這些功能進行化簡以減少延遲。

其中 *tanh* 和 *sigmoid* 使用查表的方法實作。為了減少表所占用的硬體資源，我們使用了奇函數的對稱性質，設計取樣的數量與分布以增加運算速度並減少硬體使用資源。下表以 Vitis HLS 提供的 library 與我們自己實作的做比較。

表1：Vitis HLS 提供的 library 與我們自己實作的比較
資料來源：專題組員自行繪製。

(1 cycle = 10 ns)

tanh	Latency(ns)	DSP	FF	LUT
vitis (hls_math)	370	33	2311	4700
查表版本	8.320	2	0	1160

sigmoid	Latency(ns)	DSP	FF	LUT
vitis (hls_math)	230	29	1348	4129
查表版本	8.523	2	0	1483

模型訓練結果

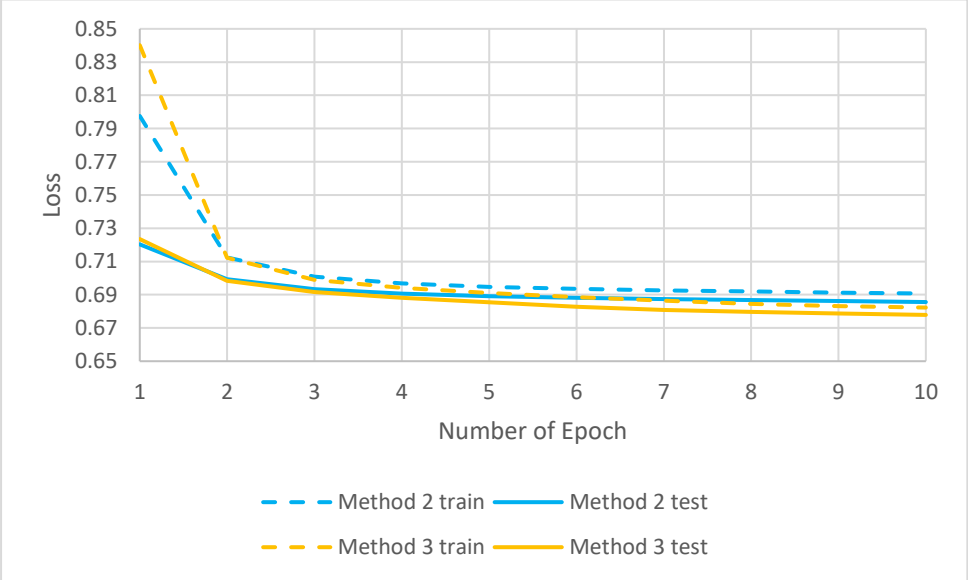

圖3：method 2 和 method 3，預測未來的流動性趨勢
資料來源：專題組員自行繪製。

從圖中可以看出兩種方法 train 跟 test 的 loss 在第二個 epoch 之後便不再有明顯變化，同時使用 order flow 相較於使用 limit order book 有更少的 loss。此外

發現，order flow 每一個 feature 的相較於包含在 orderbook 的買價與賣價擁有更小的值域範圍。因此在後續的硬體實驗中我們可以使用較少的 bit 數來區分不同的 input 值。

然而當我們進一步分析準確度時，如下圖所示，我們發現，無論是使用 orderbook 或是 order flow 作為 input，訓練完的模型總是傾向於預測所有情形為趨勢0，也就是流動性會變大。

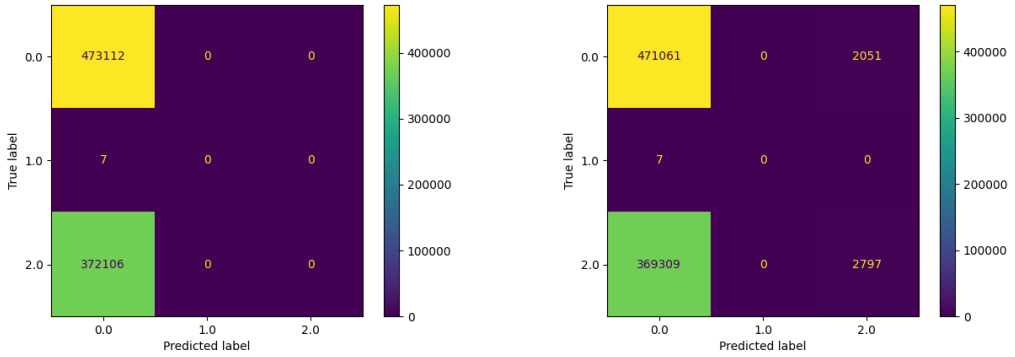

圖4：訓練完的模型總是傾向於預測所有情形為趨勢0
資料來源：專題組員自行繪製。

心得感想

這次專題研究中接觸到了 HLS、AI 和計算機網路概論的內容，學習到了很多新的知識。專題一開始什麼都不懂的時候在研究上難免會不知道該看些什麼，但隨著閱讀的東西越來越多時漸漸地知道該往那些方面研究。感謝教授每個禮拜的指導，每當遇到瓶頸教授給的建議都能給出新的方向。