智慧化企業整合

基於深度學習之

鋰離子電池健康狀態預測模型

指導教授: 邱銘傳 教授

第 9 組

109034403 伍仰輝

109034532 張郁杰

109034537 邱靖中

109034541 蘇詠心

中 華 民 國 １０９ 年 十 二 月 十 四 日

目錄

圖目錄

表目錄

1. 簡介

1.1 背景與動機

近年來，隨著電池技術的進步，以電池為主要能量供給或能量儲存的設備已被廣泛運用於各個領域，例如工業產品、計算機，電動汽車，航空航天，通訊基地台，軍事裝備等。鋰離子電池由於具有能量密度高、使用壽命長、放電率低、無污染等多項優點，成為現在電池發展的趨勢。

鋰離子電池之健康狀態 (SOH) 和剩餘壽命（RUL）預測是電池管理系統中的兩個重要因素，其評估方法的準確性將直接影響電池管理系統的性能。鋰離子電池的健康狀態（SOH）以百分比的形式呈現，表示電池的當前健康狀況。通過評估鋰離子電池的 SOH，我們可以瞭解每個電池的健康程度，以確保電源系統的穩定性。根據標準規定，當電池的健康狀態下降到 70%-80% 時，表示電池已經老化而不能繼續使用。若能夠發展預測鋰離子電池健康狀態 (SOH)，就能提前預知電池狀況，並更換達到故障閾值的電池，將可以確保電池之安全運行。

1.2 研究目的

由於實際情況鋰離子電池每次充放電之額定容量無法藉由感測器即時得出，因此本研究欲萃取歷史電池時間序列資料的特徵 (如電流、電壓、溫度等)，建立健康指標預測模型，使電池每次充放電接能夠藉由模型得出當下即時的電池健康狀態，將能夠預防電池使用過度而失效的風險。

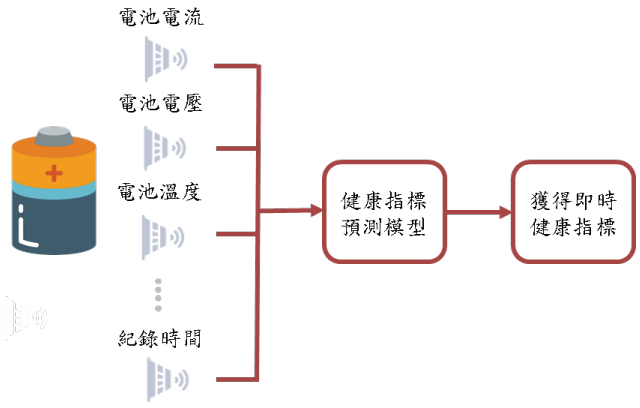


圖 1、鋰離子電池特徵參數蒐集示意圖

2. 資料集介紹

本研究使用 NASA Prognostics Data Repository “Battery Data Set” (2007) 的公開資料集 (電池編號：B0005) 進行鋰離子電池的預測，裡面共有 168 筆充電與放電數據。原始資料以 matlab 檔呈現，電池數據分為充電狀態、阻抗狀態與放電狀態。每個狀態都會記錄該狀態之連續數據，例如充電狀態紀錄

電池電流、電壓、溫度與充電器電流、充電器電壓與時間點。而放電狀態除了上述數據種類之外，另外存有容量 (Capacity) 之數據，代表該次放電所剩電池之最大容量。

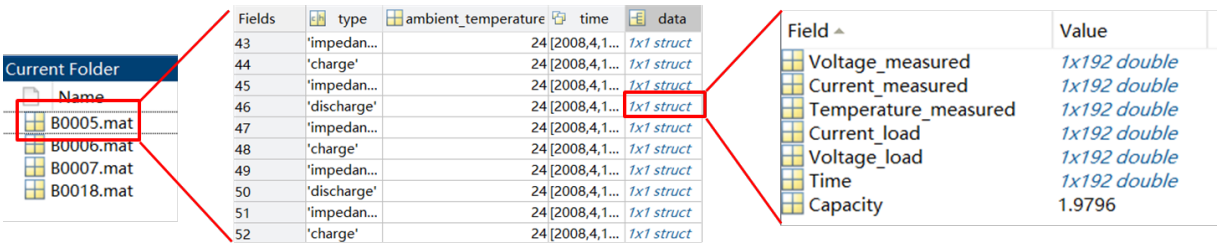


圖 2、NASA 鋰離子電池公開資料架構示意圖

由於電池各自的充電與放電數量不一致，因此本研究將總次數採以較小之次數當作循環次數。另外根據 SOH (當次充放電循環之電池容量/額定容量)來判定電池最終壽命在哪，如圖 3 所示，由於在第 124 次充放電電池的 SOH 已經趨於 70%，而第 125 次以低於標準，將判定為失效。此實驗將不考慮阻抗的因素，因為電池阻抗較偏向物理專業相關運用，因此這次只會考慮到充電與放電的資訊。

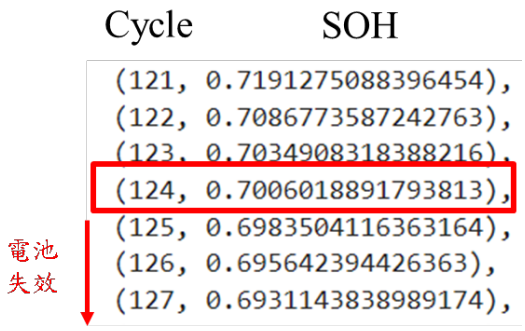


圖 3、電池健康狀態失效定義示意圖

3. 研究方法

圖 4 為研究架構圖，於資料前處理的過程會經由檔案轉換、特徵萃取、特徵篩選、標準化與資料分割等步驟，並將處理完的特徵預測電池 SOH。以下將介紹各方法之定理。

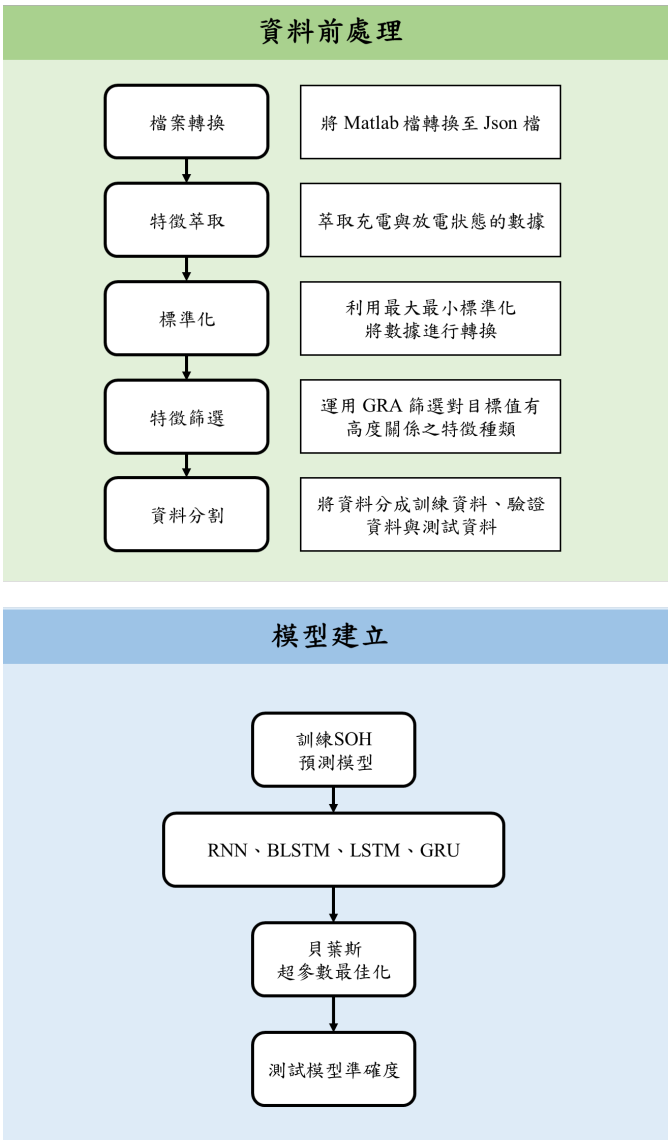


圖 4、研究架構圖

3.1 資料前處理

3.1.1 灰色關聯分析

灰色關聯分析簡稱 GRA，為一分析方式去尋找系統中各因素與目標之間的關聯，並給予關聯係數代表因素與目標之間的關聯程度。通常定義最高的關聯係數為 1，最低為 0。

3.1.2 最大最小標準化

最大最小標準化之用意在於將資料等比例縮放到 [0,1] 區間中，詳細公式如下。

$$X_{nom} = \frac{X - X_{min}}{X_{max} - X_{min}} \in [0,1]$$

3.2 深度學習演算法

3.2.1 遞歸神經網絡（RNN）

遞歸神經網絡（RNN）是專門設計用於處理時間序列數據的模型，適用於文本，音節，視頻，天氣預報數據，股票交易以及其他時間序列數據。與傳統的神經網絡不同，它的核心概念是強調數據與數據之間的關係，就像網路具有暫存區一樣，它將根據過去的記憶確定當前的輸出。簡單來說，神經網絡只記住最近的事件，而較早的事件已被遺忘。

3.2.2 長短期記憶（LSTM）

長短期記憶（LSTM）是一種特殊的 RNN 模型。與 RNN 相比，LSTM 可以更好地處理長期順序數據，並通過存儲功能解決長期依賴的問題。與常規 RNN 相比，LSTM 更為複雜。它還具有三個控制門，即輸入門，輸出門和忘記門。當將值寫入存儲單元時，它具有一個存儲單元，必須通過輸入門並且只有在門打開時才能寫入該值。至於輸入門是否打開，LSTM 會自己學習，輸出門則控制是否可以讀取存儲單元的值，而遺忘門決定何時應清除存儲單元的值。

3.2.3 門控循環單元（GRU）

門控循環單元（GRU）是 RNN 的門控機制。 GRU 可以在某些較小和較不頻繁的頻率下顯示更好的性能數據集。它用更新門替換 LSTM 中的忘記門和輸入門，並合併單元狀態和隱藏狀態以計算新信息。

3.3 超參數優化 (Hyperparameter tuning)

3.3.1 手動式調整

本次調整的超參數優化有啟動函數 (Activation Function)，隱形層 (Hidden Layer)，神經元數 (Number of neuron)，捨棄率 (Dropout rate)，學習率 (Learning rate) 等，詳細內容如表 C。

Parameter	Value
Parameter optimizer	Adam
Loss function	MSE
Activation function	ReLU, Softmax, Leaky ReLU
Batch size	{10, 16, 32}
Epochs	500
Dropout rate	{0.00, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10,

	0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.20}
Number of hidden layers	{1 ,2 ,3}
Number of neurons	{16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256}
Learning rate	{0.01, 0.001, 0.0001}

由以上可見，調整的參數組合共有 C(1,1)*C(1,1) *C(3,1) *C(3,1) *C(1,1) *C(20,1) *C(3,1) *C(16,1) *C(3,1) = 25,920 種組合，若是以傳統手動調超參數，會沒有辦法完全嘗試所有的組合，也有可能無視交互作用，因此最佳化演算法更扮演著重要的角色。

3.3.2 貝葉氏最佳化 (Bayesian Optimization)

為了達到更佳的準確率，深度學習演算法必須進行超參數優化，通常都是靠人工試錯的方式找到"最優"超參數。但是這種方式效率太慢，所以相繼提出了網格搜索 (Grid Search, GS) 和隨機搜索 (Random Search,RS)。但其實這兩個既耗費長時間在訓練模型也無法透過之前套參數來改善下一次對超參數的選擇，表示網格搜索和隨機搜索浪費了很多時間和資源去評估一個爛的超參數，所以有研究提出貝葉氏最優化 (Bayesian Optimization)。貝葉氏最優化建立一個含有目標函數的機率模型，並且使用它去選擇最佳的超參數去評估正確的目標函數，其概念類似條件機率。

$$x^\star = \arg\min_{x \in \mathcal{X}} f(x)$$

處理這個目標函數必須包含以下 4 個部分：

1. 目標函數: 當然必須要有目標函數, 這是我們必須最小化的部分, 降低驗證錯誤透過超參數的調整(tuning)

2. 範圍:我們必須明確確定參數的範圍，因為有可能 A 這個超參數只能為整數(integer)，而超參數可以為浮點數(float)來表達，有些則因為是分類學數據(categorical data)，所以只能有 0，1 來表示那個群體

3. 最佳化演算法: 方法可以用來建立條件模型包含超參數和誤差項

4. 結果:必須紀錄當你使用超參數和誤差項在目標函數的所有結果

本次的研究也是透過貝葉氏最佳化，主要目的與大多數機器學習算法相似。學習模型的表達形式是在一定範圍內找到函數的最大值(最小值)。貝葉氏最佳化是一種解決最佳化問題的技術，通過一些耗時的操作進行評估，但是可以將模型擬合到相對適當的觀測值，並使用該模型來預測參數空間。透過貝葉氏最佳化會產出 10 組最佳的組合，再套入模型後選出最佳的演算法。

3.4 評估指標

3.4.1 均方根誤差 (RMSE)

均方根誤差 (RMSE) 公式如下：

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

3.4.2 平均絕對百分比誤差 (MAPE)

平均絕對百分比誤差 (MAPE) 公式如下：

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| X100\%$$

3.4.3 平均絕對誤差 (MAE)

平均絕對誤差 (MAE) 公式如下：

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

4. 實驗流程與結果

4.1 資料前處理

圖 5 為資料前處理架構圖，詳細步驟將依序說明：

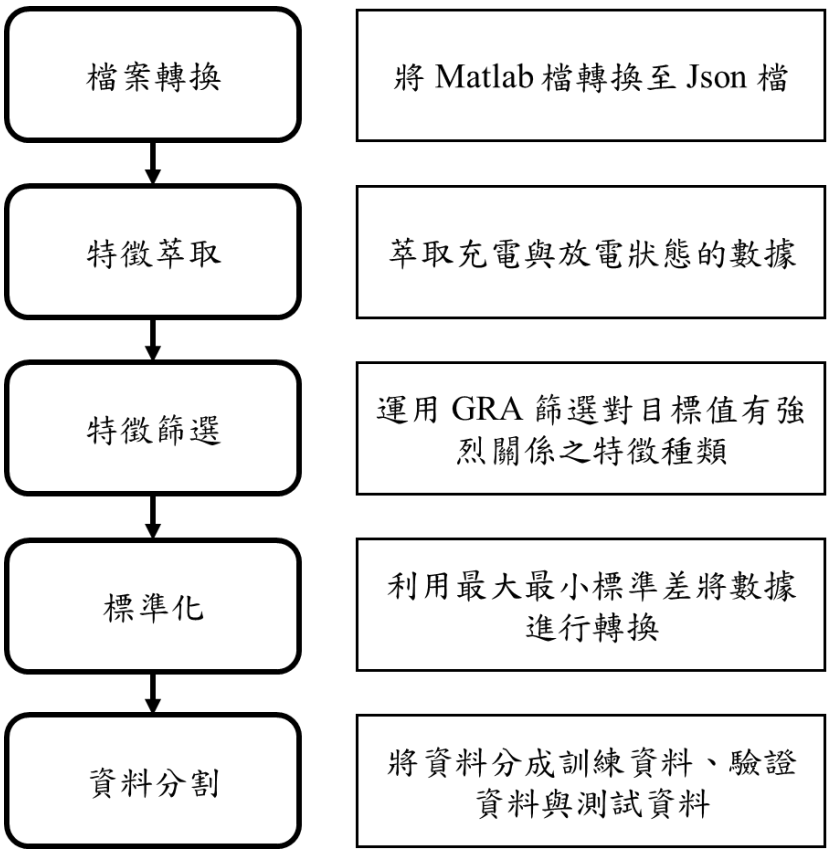


圖 5、資料前處理流程圖

4.1.1 檔案轉換

由於原始資料為 matlab 檔，必須經由轉換的方式轉換成 json 的架構方便後續 python 資料萃取的作業，如圖 6，而檔案轉換程式碼示意圖如圖 7。

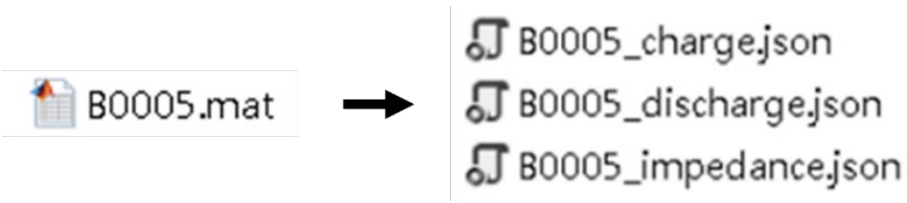


圖 6、檔案轉換示意圖

```
#轉換資料
def build_dictionaries(mess):

    discharge, charge, impedance = {}, {}, {}

    for i, element in enumerate(mess):

        step = element[0][0]

        if step == 'discharge':
            discharge[str(i)] = {}
            discharge[str(i)]["ambient_temperature"] = str(element[1][0][0])
            year = int(element[2][0][0])
            month = int(element[2][0][1])
            day = int(element[2][0][2])
            hour = int(element[2][0][3])
            minute = int(element[2][0][4])
            second = int(element[2][0][5])
            millisecond = int((second % 1)*1000)
            date_time = datetime.datetime(year, month, day, hour, minute, second, millisecond)

            discharge[str(i)]["date_time"] = date_time.strftime("%d %b %Y, %H:%M:%S")

            data = element[3]

            discharge[str(i)]["Voltage_measured"] = data[0][0][0][0].tolist()
            discharge[str(i)]["Current_measured"] = data[0][0][1][0].tolist()
            discharge[str(i)]["Temperature_measured"] = data[0][0][2][0].tolist()
            discharge[str(i)]["Current_load"] = data[0][0][3][0].tolist()
            discharge[str(i)]["Voltage_load"] = data[0][0][4][0].tolist()
            discharge[str(i)]["Time"] = data[0][0][5][0].tolist()
            discharge[str(i)]["Capacity"] = float(data[0][0][6][0][0])

        if step == 'charge':
            charge[str(i)] = {}
            charge[str(i)]["ambient_temperature"] = str(element[1][0][0])
            year = int(element[2][0][0])
            month = int(element[2][0][1])
            day = int(element[2][0][2])
            hour = int(element[2][0][3])
            minute = int(element[2][0][4])
            second = int(element[2][0][5])
            millisecond = int((second % 1)*1000)
            date_time = datetime.datetime(year, month, day, hour, minute, second, millisecond)

            charge[str(i)]["date_time"] = date_time.strftime("%d %b %Y, %H:%M:%S")

            data = element[3]
```

圖 7、檔案轉換程式碼示意圖

4.1.2 特徵萃取

我們能夠從電池的電流、電壓與溫度的資訊的變化，瞭解電池充電與放電的情況，當電池在充電的時候，電壓會持續升高而電流會維持定值，等到電壓上升到一定的程度會停在某個定值，此時的電流則會下滑。而當電壓開始下滑可以視為電池正在放電，這時的電流則會維持在一定的負值，電流電壓趨勢圖如圖 8。

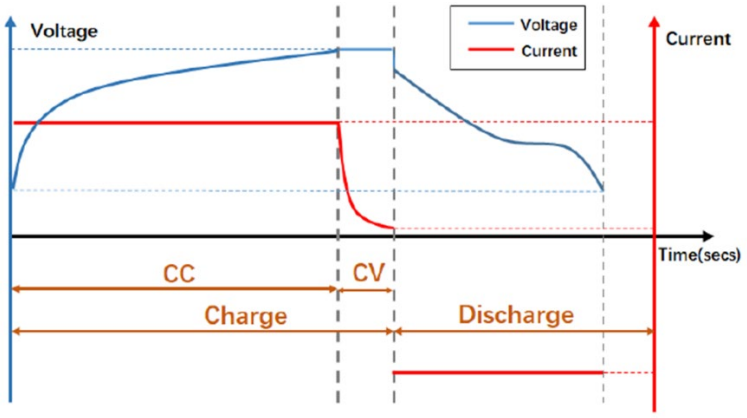
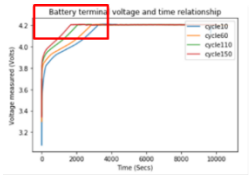


圖 8、充放電狀態電流與電壓趨勢圖

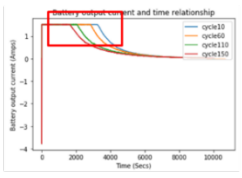
（資料來源：Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach (2018)）

本實驗以此定理擷取各循環中每種數據充放電結束的瞬間數值與其紀錄時間。如圖 9 所示，第一張子圖為充電狀態電池電壓的趨勢，不同顏色的線代表不同循環次數，可以發現隨著充放電次數的增加，其趨勢圖會有時間評移的情況，因此我們將擷取每次循環之電壓最早穩定的值與紀錄時間當作其中兩項特徵值，其他特徵的擷取方式如電流、溫度與放電狀態之所有數據

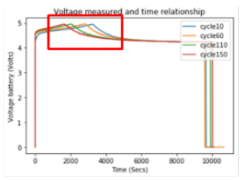
種類以此類推，放電狀態特徵擷取示意圖如圖 10。



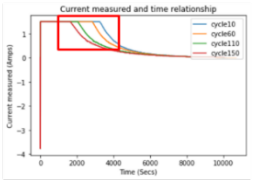
電池電壓與累計時間關係圖



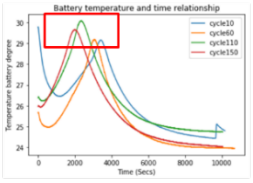
電池電流與累計時間關係圖



充電器電壓與累計時間關係圖

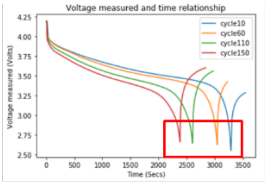


充電器電流與累計時間關係圖

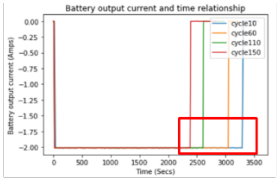


電池溫度與累計時間關係圖

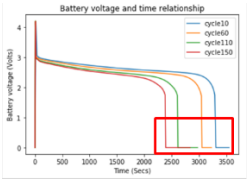
圖 9、充電狀態數據擷取示意圖



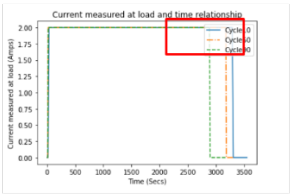
電池電壓與累計時間關係圖



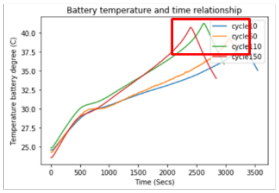
電池電流與累計時間關係圖



負載電壓與累計時間關係圖



負載電流與累計時間關係圖



電池溫度與累計時間關係圖

圖 10、放電狀態數據擷取示意圖

所有的特徵種類可以歸類於表 1。

表 1、特徵種類總表

排序	特徵名稱
1	Charge_Voltage_Measured
2	Charge_Voltage_Measured_Time
3	Charge_Current_Measured
4	Charge_Current_Measured_Time
5	Charge_Temperature_Measured
6	Charge_Temperature_Measured_Time

7	Charge_Current_Charger
8	Charge_Current_Charger_Time
9	Charge_Voltage_Charger
10	Charge_Voltage_Charger_Time
11	Discharge_Voltage_Measured
12	Discharge_Voltage_Measured_Time
13	Discharge_Current_Measured
14	Discharge_Current_Measured_Time
15	Discharge_Temperature_Measured
16	Discharge_Temperature_Measured_Time
17	Discharge_Current_Load
18	Discharge_Current_Load_Time
19	Discharge_Voltage_Load
20	Discharge_Voltage_Load_Time

4.1.3 標準化

為了減少數據的跨度，以最大最小標準化調整數據，以利後續模型的預測準確度，標準化結果圖如圖 11。

	Charge_Voltage_measured_Time	Charge_Voltage_measured	Charge_Current_measured_Time	Charge_Voltage_measured
0	0.000000	0.435527	0.000000	0.364566
1	0.439930	0.951665	0.996960	0.397478
2	0.467502	1.000000	1.000000	0.334085
3	0.423650	0.898847	0.983559	0.390953
4	0.444654	0.998646	0.974484	0.324410
...	...	...	...	...
163	0.006054	0.099706	0.085424	0.407633
164	0.050286	0.026810	0.082707	0.339012
165	0.047151	0.025505	0.078238	0.652572
166	0.032086	0.016743	0.075610	0.320273
167	0.005554	0.047570	0.091502	0.295710

Discharge_Voltage_charge_Time	Discharge_Voltage_charge	Capacity_Residual
1.000000	0.550562	0.999706
0.982418	0.460674	0.999903
0.963578	0.764045	1.000000
0.963866	0.494382	0.999994
0.961894	0.303371	0.999883
...	...	...
0.010330	0.303371	0.000433
0.001137	0.426966	0.000987
0.000000	0.325843	0.001758
0.037545	0.415730	0.002742
0.065063	0.286517	0.003931

圖 11、資料標準化執行結果

4.1.4 特徵篩選

本實驗針對 SOH 預測，於模型建立之前先以 GRA 進行特徵篩選，找出和 SOH 關聯最大的特徵種類，GRA 程式碼如圖 12，而實驗結果如表 2，可以發現放電狀態之電池電壓紀錄時間、放電狀態之電池電流紀錄時間、放電狀態之電池溫度紀錄時間與放電狀態之負載電壓紀錄時間這四種特徵得到相對高分。因此以這四種數據當作 SOH 預測的特徵。

```
def Gray(dataset):

    gray=dataset

    gray=(gray - gray.min()) / (gray.max() - gray.min())
    gray=gray.fillna(method='bfill')
    gray=gray.fillna(method='ffill')


    std=gray.iloc[:,20]#標準
    ce=gray.iloc[:,0:20]#比較

    n=ce.shape[0]
    m=ce.shape[1]

    a=zeros([m,n])
    for i in range(m):
        for j in range(n):
            a[i,j]=abs(ce.iloc[j,i]-std[j])

    c=amax(a)
    d=amin(a)


    result=zeros([m,n])
    for i in range(m):
        for j in range(n):
            result[i,j]=(d+0.5*c)/(a[i,j]+0.5*c)

    #求均值得到灰色關聯值
    result2=zeros(m)
    for i in range(m):
        result2[i]=mean(result[i,:])
    RT=pd.DataFrame(result2)
```

圖 12、GRA 程式碼示意圖

表 2、GRA 執行結果

Feature	關聯係數
Charge_Voltage_Measured	0.5629
Charge_Voltage_Measured_Time	0.6795
Charge_Current_Measured	0.6031
Charge_Current_Measured_Time	0.7212
Charge_Temperature_Measured	0.6135
Charge_Temperature_Measured_Time	0.6270
Charge_Current_Charger	0.5767
Charge_Current_Charger_Time	0.9677
Charge_Voltage_Charger	0.6522
Charge_Voltage_Charger_Time	0.7218
Discharge_Voltage_Measured	0.5672
Discharge_Voltage_Measured_Time	**0.9928**
Discharge_Current_Measured	0.6462
Discharge_Current_Measured_Time	**0.9865**
Discharge_Temperature_Measured	0.5035
Discharge_Temperature_Measured_Time	**0.9886**
Discharge_Current_Load	0.4685

Discharge_Current_Load_Time	0.6713
Discharge_Voltage_Load	0.6675
Discharge_Voltage_Load_Time	**0.9928**

4.1.5 資料分割

模型建立之前，需要將資料進行分割的動作，本實驗將資料分成訓練資料 80%、驗證資料 10%、測試資料 10%。

4.2 模型建立

圖 13 為 SOH 模型建立架構圖，RUL 模型建立架構圖與此相同，並運用 RNN、LSTM 與 GRU 等不同深度學習演算法建立模型。

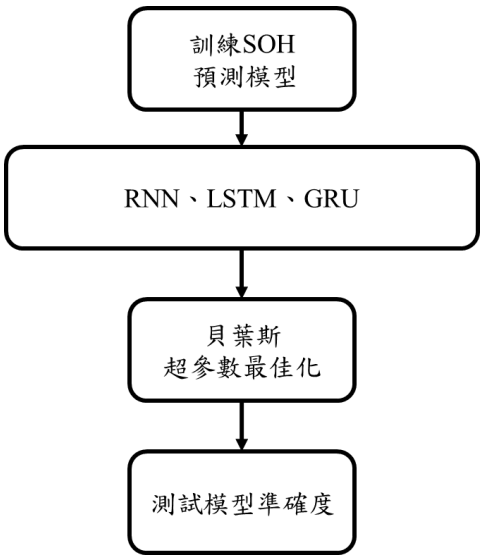


圖 13、模型建立流程圖

本實驗之模型架構如圖 14，以 RNN 模型為例，預設參數設定為 batch size =10、第一層神經元數為 64、第二層神經元數為 128、第三層神經元數為 256、第一層 Dropout 為 0.02、第二層 Dropout 為 0.07、各層 Activation Function 為 ReLU、Optimizer 為 Adam、學習率(Learning Rate)為 0.001。LSTM 與 GRU 模型如同此架構進行實作。

```
n_batch = 10
model_RNN_01 = Sequential()
model_RNN_01.add(SimpleRNN(64, batch_input_shape=(n_batch, X_train0.shape[1], X_train0.shape[2])))
model_RNN_01.add(Dense(128, activation='relu'))
model_RNN_01.add(Dropout(0.02))
model_RNN_01.add(Dense(256, activation='relu'))
model_RNN_01.add(Dropout(0.07))
model_RNN_01.add((Dense(1)))
adam = keras.optimizers.Adam(lr=0.001)
model_RNN_01.compile(loss='mean_squared_error', optimizer = adam)
history = model_RNN_01.fit(X_train0, y_train0,epochs=500,validation_data=(X_valid, y_valid), verbose=2, shuffle=False)
print(rmse(allprevalue,alltruevalue),mape(allprevalue,alltruevalue),'%',mae(allprevalue,alltruevalue))
```

```
0.0019860728160747748 0.19057215485740878 % 0.001632483698051817
```

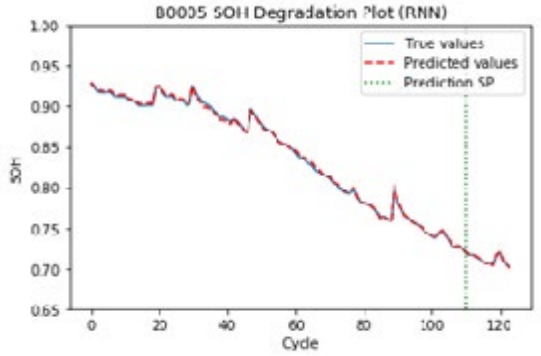


圖 14、模型架構程式碼示意圖

參數調整的過程，首先我們先比較 Softmax、ReLU 與 Leaky ReLU 等多種 Activation Function 去訓練模型，並以測試資料集去評估模型，觀察哪一種效果較為良好，而評估指標將採用 RMSE、MAPE 與 MAE。接著再比較 Batch Size，而在比較過程中其餘參數皆固定為預設值。調整完 Activation Function 與 Batch Size 後，我們以貝葉氏最佳化的方式調整各層神經元數、Dropout 的比率還有學習率等其餘參數。貝葉氏最佳化結果示意圖如圖 15，此參數最佳化套件會列出前十個最佳調整結果。接著依序介紹 RNN、LSTM 與 GRU 的參數調整過程。

圖 15、貝葉氏超參數最佳化結果示意圖

4.2.1 RNN 模型參數調整過程

以 RNN 模型評估指標來觀察，可以知道三者 Activation Function 中 ReLU 表現最為良好，因此模型將採取 ReLU 當作 RNN 模型的 Activation Function。RNN 模型之 Activation Function 評估比較表如表 3。

表 3、RNN 模型之 Activation Function 評估比較表

Activation Function	RMSE	MAPE	MAE	Selection
Softmax	0.0175	1.81%	0.0143	
ReLU	0.0043	0.45%	0.0038	V
Leaky ReLU	0.0078	0.78%	0.0067	

接著調整 Batch Size 的參數，我們比較四種參數設定如 16 和 32，結果發現在 Batch Size 為 50 的時候模型表現較好，因此保留其設定。RNN 模型之 Batch Size 評估比較表如表 4。

表 4、RNN 模型之 Batch Size 評估比較表

Batch Size	RMSE	MAPE	MAE	Selection
10	0.0043	0.45%	0.0038	
16	0.0050	0.51%	0.0040	
32	0.0020	0.28%	0.0020	V

調整完 Activation Function 和 Batch Size 後，以 ReLU 和 Batch Size=32，執行貝葉氏最佳化調整其餘參數。而貝葉氏最佳化參數設定如表 5。

表 5、RNN 模型之貝葉氏最佳化結果參數設定表

超參數	參數設定
Activation Function	ReLU
Batch Size	32
第一層神經元數	224
第二層神經元數	224
第三層神經元數	16
第一層 Dropout	0.08
第二層 Dropout	0.01
學習率	0.001

最後再以貝葉氏得出來的參數設定重新訓練模型，可以從三種評估指標看出準確度的進步。

表 6、RNN 模型之使用貝葉氏超參數最佳化模型評估結果

評估指標	貝葉氏最佳化結果
RMSE	0.0019
MAPE	0.19%
MAE	0.0016

4.2.2 LSTM 模型參數調整過程

以 LSTM 模型評估指標來觀察，可以知道三者 Activation Function 中 ReLU 表現最為良好，因此模型將採取 ReLU 當作 LSTM 模型的 Activation Function。LSTM 模型之 Activation Function 評估比較表如表 7。

表 7、LSTM 模型之 Activation Function 評估比較表

Activation Function	RMSE	MAPE	MAE	Selection
Softmax	0.0124	1.63%	0.0227	
ReLU	0.0019	0.25%	0.0035	V
Leaky ReLU	0.0054	0.71%	0.0099	

接著調整 Batch Size 的參數，我們比較四種參數設定如 16 和 32，結果發現在 Batch Size 為 32 的時候模型表現較好，因此保留其設定。RNN 模型

之 Batch Size 評估比較表如表 8。

表 8、LSTM 模型之 Batch Size 評估比較表

Batch Size	RMSE	MAPE	MAE	Selection
10	0.0019	0.25%	0.0035	
16	0.0048	0.65%	0.0047	
32	0.001	0.13%	0.0009	V

調整完 Activation Function 和 Batch Size 後，以 ReLU 和 Batch Size=32，執行貝葉氏最佳化調整其餘參數。而貝葉氏最佳化參數設定如表 9。

表 9、LSTM 模型之貝葉氏最佳化結果參數設定表

超參數	參數設定
Activation Function	ReLU
Batch Size	32
第一層神經元數	128
第二層神經元數	240
第三層神經元數	96
第一層 Dropout	0.07
第二層 Dropout	0.09
學習率	0.001

最後再以貝葉氏得出來的參數設定重新訓練模型，可以從三種評估指標看出準確度的進步。

表 10、LSTM 模型之使用貝葉氏超參數最佳化模型評估結果

評估指標	貝葉氏最佳化結果
RMSE	0.0004
MAPE	0.05%
MAE	0.0004

4.2.3 GRU 模型參數調整過程

以 GRU 模型評估指標來觀察，可以知道三者 Activation Function 中 ReLU 表現最為良好，因此模型將採取 ReLU 當作 GRU 模型的 Activation Function。GRU 模型之 Activation Function 評估比較表如表 11。

表 11、GRU 模型之 Activation Function 評估比較表

Activation Function	RMSE	MAPE	MAE	Selection
Softmax	0.0192	1.92%	0.0192	

ReLU	0.0068	0.68%	0.0068	V
Leaky ReLU	0.0124	1.24%	0.0124	

接著調整 Batch Size 的參數，我們比較四種參數設定如 16 和 32，結果發現在 Batch Size 為 32 的時候模型表現較好，因此保留其設定。RNN 模型之 Batch Size 評估比較表如表 12。

表 12、GRU 模型之 Batch Size 評估比較表

Batch Size	RMSE	MAPE	MAE	Selection
10	0.0068	0.68%	0.0068	
16	0.0051	0.59%	0.0043	
32	0.0029	0.40%	0.0029	V

調整完 Activation Function 和 Batch Size 後，以 ReLU 和 Batch Size=32，執行貝葉氏最佳化調整其餘參數。而貝葉氏最佳化參數設定如表 13。

表 13、GRU 模型之貝葉氏最佳化結果參數設定表

超參數	參數設定
Activation Function	ReLU
Batch Size	32
第一層神經元數	80
第二層神經元數	96
第三層神經元數	224
第一層 Dropout	0.02
第二層 Dropout	0.01
學習率	0.001

最後再以貝葉氏得出來的參數設定重新訓練模型，可以從三種評估指標看出準確度的進步。

表 14、GRU 模型之使用貝葉氏超參數最佳化模型評估結果

評估指標	貝葉氏最佳化結果
RMSE	0.0009
MAPE	0.13%
MAE	0.0009

4.3 實驗結果

以下依序列出個模型最佳預測 SOH 之評估結果，可以發現預測 SOH 最佳的演算法為 LSTM，表 15 為三種模型各自的評估指標最佳結果，而圖 16 為 LSTM 模型預測視覺圖，以目測可以看出預測值和實際值接近一致，說明預測效果良好。

表 15、三種之評估指標結果表

	RNN	LSTM	GRU
RMSE	0.0019	0.0004	0.0009
MAPE	0.19%	0.05%	0.13%
MAE	0.0016	0.0004	0.0009

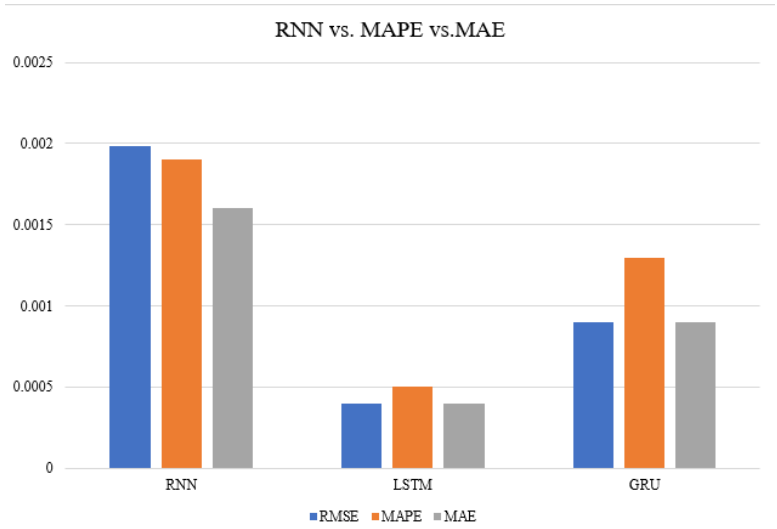


圖 16、評估指標結果圖

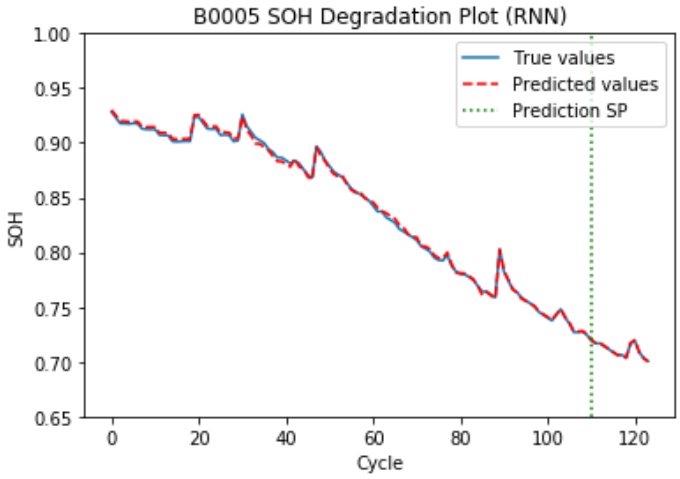


圖 17、RNN 模型預測視覺圖

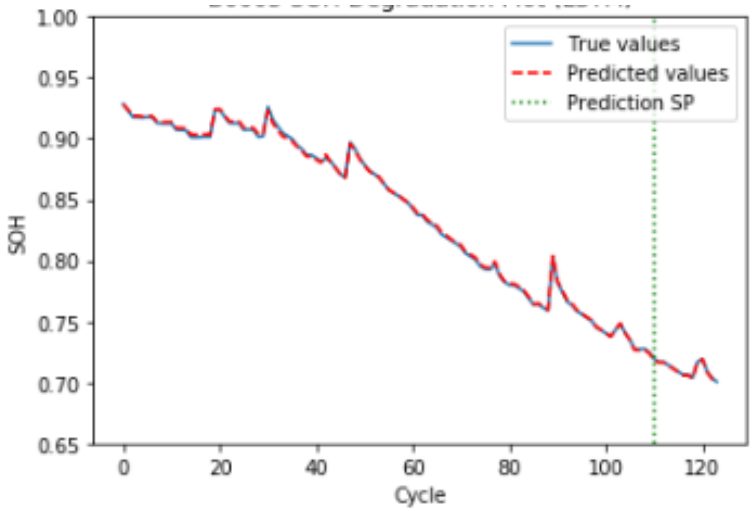


圖 18、LSTM 模型預測視覺圖

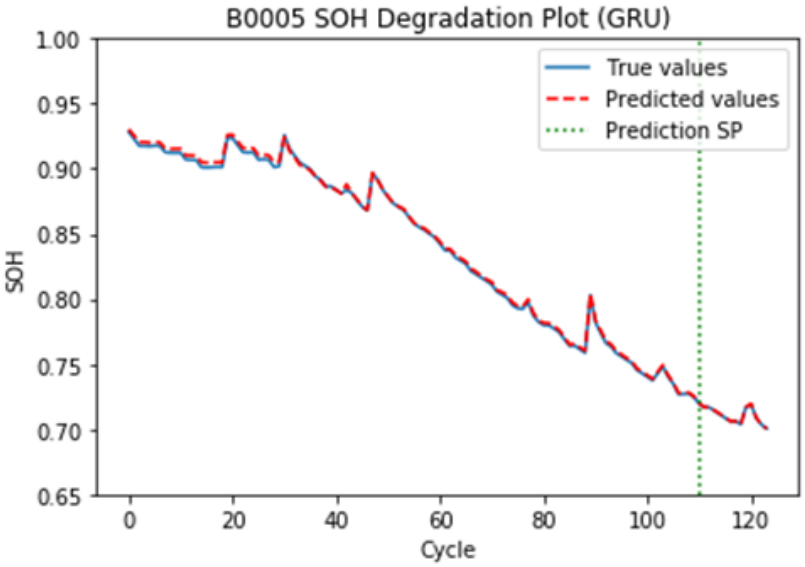


圖 19、GRU 模型預測視覺圖

以下圖表可以看出若無使用貝葉氏最佳化的結果與使用貝葉氏最佳化後的結果的差異，證明了超參數優化在尺深度學習扮演者重要的角色。即使沒有貝葉氏最佳化，LSTM 都依然是較佳的預測模型。

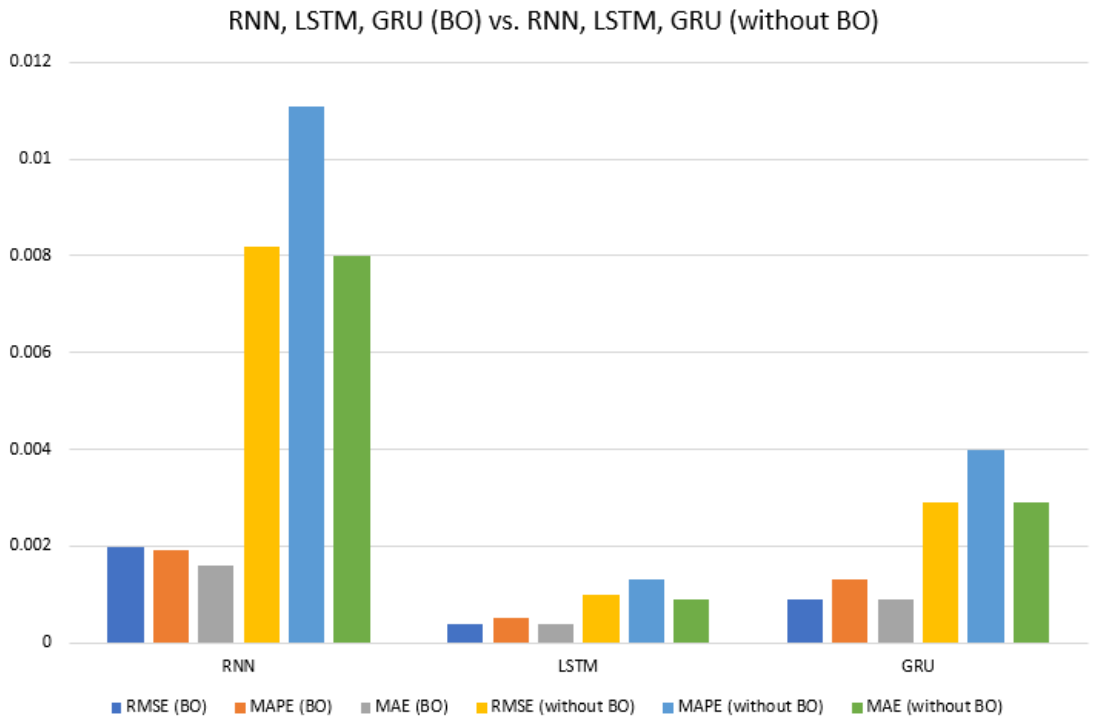


圖 20、有無貝葉氏結果比較圖

5. 結論

這次的實驗運用不同的方式進行資料前處理，像是 GRA 的特徵篩選，與最大最小標準化之數據跨度調整，而預測模型我們採用 RNN、LSTM 與 GRU 三種深度學習演算法預測 NASA 鋰離子電池健康狀態並利用貝葉氏最佳化條找出最好的參數設定，並且於實驗的最後得知 LSTM 模型對於 SOH 有良好的預測效果。

於實務上，由于鋰離子的健康狀況在使用上是很難再用儀器測量的，因此一般都會到電池壞了才進行更換，假設我們預知電池的健康狀態，就可以避免電池損壞所導致的意外，也顯示出預測鋰離子電池健康狀況的重要性。

此實驗可以當作鋰離子電池健康況狀態評估的模擬，於研究上，此實驗的結果可以當作後續研究者參考之用。

5.1 未來展望

由於本次實驗只單純在做鋰離子電池的健康狀況，雖然 Nasa 的定義裡說明了當 SOH 達到 70% 以下的時候則代表電池健康程度極差，但是卻沒有明確的說明還可以循環充電幾次，因此未來可以針對電池在 SOH 達到不健康水準下的電池剩餘壽命 (Remaining Useful Life, RUL)。

參考資料

1. Ren, L., Zhao, L., Hong, S., Zhao, S., Wang, H., & Zhang, L. (2018). Remaining useful life prediction for lithium-ion battery: A deep learning approach. IEEE Access, 6, 50587-50598.