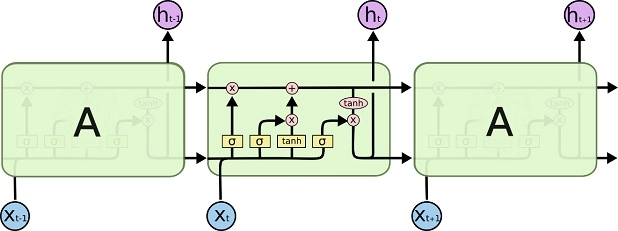
方法:

LSTM是一種時間遞歸神經網路，是何處理及預測時間序列相關事件。由LSTM建構的人工神經網絡模型已經被應用於許多方面上，例如不分段連續手寫識別及語言識別上。自1997年首先被Hochreiter and Schmidhuber [1]引入後，已經被許多人改善及變種。

而LSTM與一般的RNN差別在於，LSTM更擅長於長時間的資料之間的關係，因為LSTM具有三個類似遺忘層的部分，藉由這層0到1之間的輸出來決定資料存留多少，並保護神經單元的狀態。在一般的RNN中，隨著資料序列的長度越長，距離越遠的資料之間的關係越不容易被RNN所察覺，然而LSTM比RNN多了遺忘層的機制，可以取決每筆資料存留下的比重，也更容易可以學習時間距離長的資料之間的關係。



Pros and Cons of LSTM:

Advantages

* 1. LSTM network is universal in the sense that given enough network units it can compute anything a conventional computer can compute
  2. LSTM network is better-suited to learn from experience to classify, process and predict time series or other sequence modeling tasks than traditional RNN.
  3. Many works of handwriting recognition were done by RNN with LSTM.

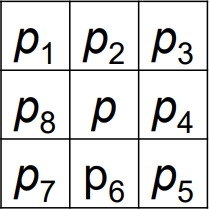
Disadvantages

1. It takes more computing time.
2. Some works have shown that RNN with rectified linear unit with appropriate initialization of parameters could achieve competitive performances with RNN LSTM on some tasks.

前處理:

Skeleton:

利用Zicheng Guo與 Righard W. Hall於1989年發表的”Parallel Thinning with Two-Subiteration Algorithms”[2] 中提出了Guo-Hall 骨架演算法快速將每張手寫數字的圖像進行細化。先對每張圖的像素進行如圖\_\_定義：為P點的四相鄰點，而為P點的對角點。



圖\_\_

為在八相鄰區域中分裂的區塊：

則是用來偵測該點是否為終端點：

其中的為：

再透過以下的判斷後，將被標記為｢可去除｣的像素改為灰階值0：

1. (奇數次迴圈)  
   (偶數次迴圈)

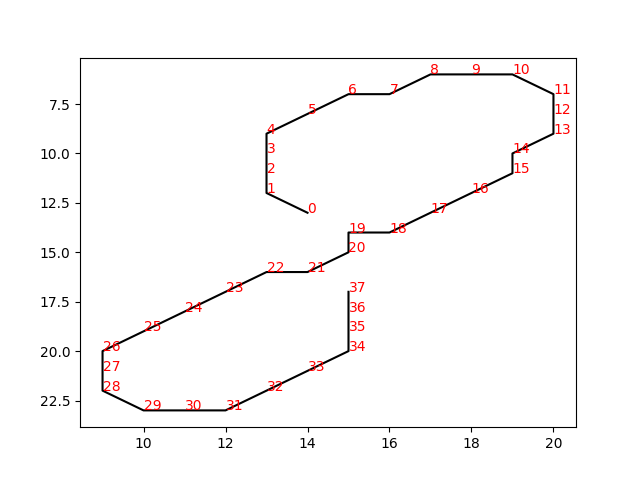
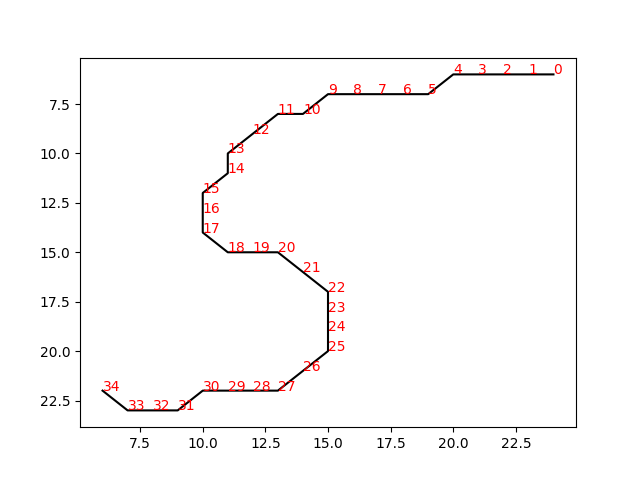
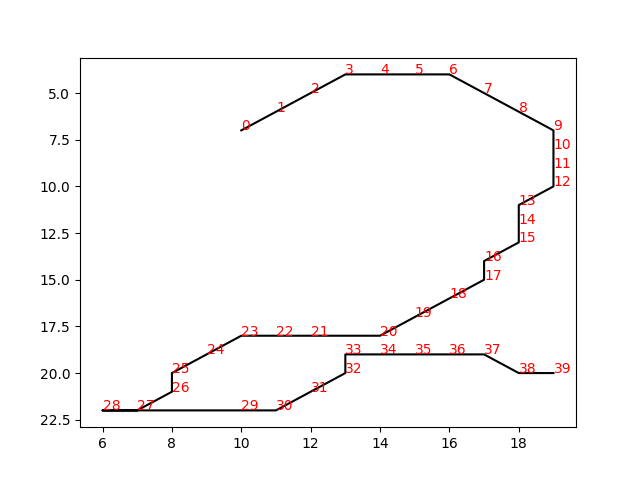
C(P) = 1是為了避免相連的區域被斷開，代表P是周圍多個區域的連接點，若P點被去除，周圍區域則會因此被斷開成為多個區塊，因此C(P)不能等於1。而為了不能讓端點區域在每次的迴圈中縮短，同時有效找出可去除的點，因此我們設定，若，代表該點為終端點。藉由迴圈將圖中的數字進行細化，當該次迴圈無法再找出可被去除的像素時，就跳出迴圈並輸出完整細化的手寫數字圖(圖\_\_)。

C:\Users\dungd\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2_233.jpg C:\Users\dungd\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5_273.jpg C:\Users\dungd\AppData\Local\Microsoft\Windows\INetCache\Content.Word\8_273.jpg

圖\_\_

Travelling Salesman Problem(TSP)

旅行推銷員問題(TSP)是一個局部最佳化問題，藉由一系列的位置座標及之間的距離關係，尋找運用最短距離完成通過每個位置一次的目標。在本次題目中，我們使用了python Suboptimal Travelling Salesman Problem(TSP) solver來幫我們完成運算最短距離問題，以輸入一個序列的點座標來運算連線最短距離作為人類手寫數字的筆畫順序(圖\_\_)。



圖\_\_

Python Tensorflow

1. 先將前處理好的筆劃資料用python讀入，並搜尋筆畫步數最多者作為每筆資料輸入的長度，並將所有資料進行整理及補0至同一長度。
2. 分批資料，用tensorflow 內建的tf.train.batch來進行之後資料批次輸入資料的作業。
3. 架設LSTM 神經元架構，設定一層的LSTM神經元及輸出層10個特徵值代表10種數字。
4. 在輸入資料至神經元前，為了讓神經元辨別出資料的時間順序，我們將資料的形狀從Batch\_number x n\_input x sequence\_length 變成Batch\_number x n\_input，將每個時間點上的資料切割出來分別送進神經元中進行訓練。
5. 將回傳的預測與真實training data set的標記比對後再回去更新神經元。
6. 42000筆資料，我們每1000筆資料分為一個批次，總共42個批次作為全部輸入的資料，並訓練40次。

[1] Hochreiter, Sepp, and Jürgen Schmidhuber. "LSTM can solve hard long time lag problems." *Advances in neural information processing systems*. 1997.

[2] Guo, Zicheng, and Richard W. Hall. "Parallel thinning with two-subiteration algorithms." *Communications of the ACM* 32.3 (1989): 359-373.