智慧型系統

期末報告

A1035501 孫茂勛

CONTENTS

1 文獻回顧

2 資料分析

3 程式設計

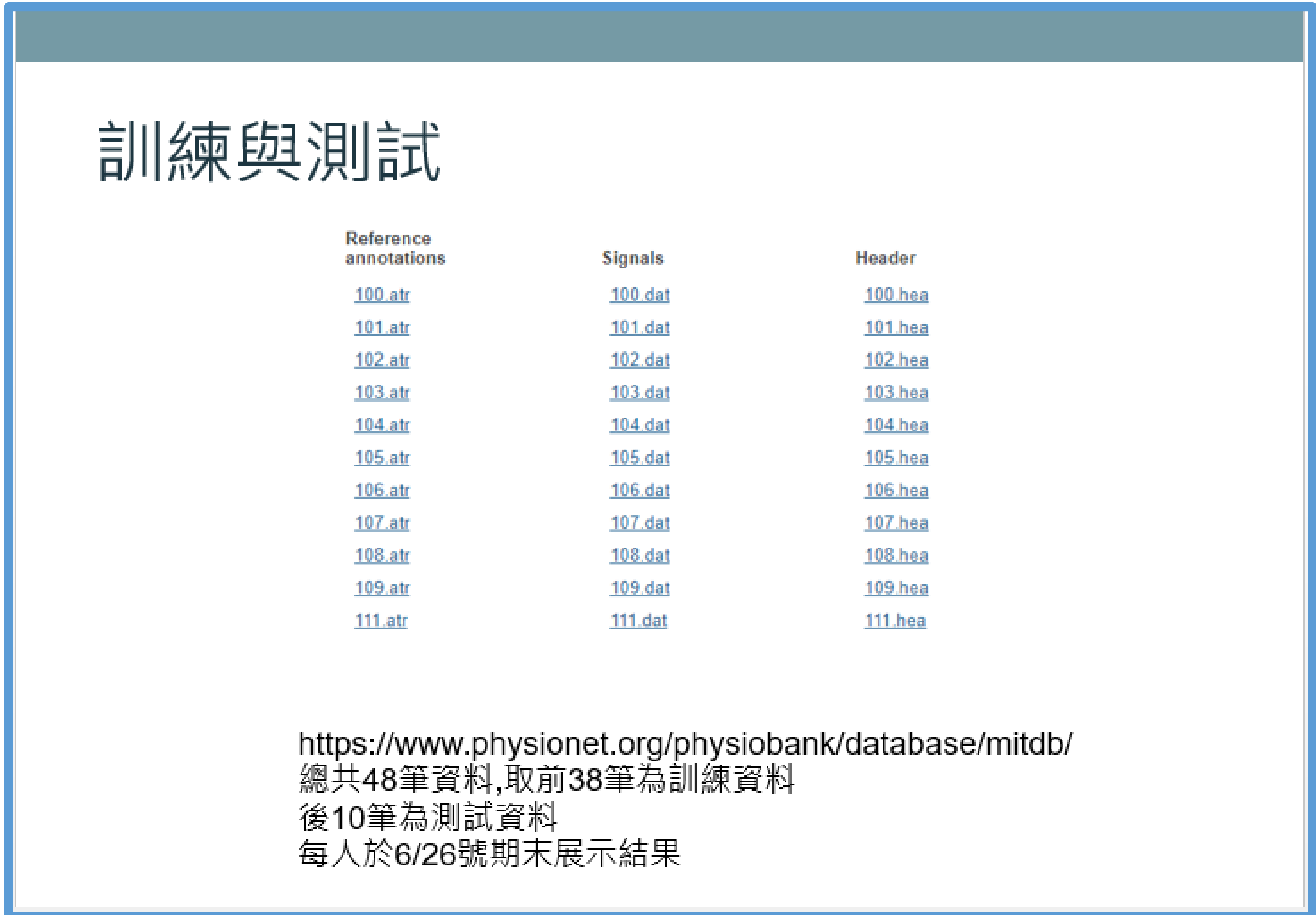
4 結論

PART ONE

文獻回顧

Project Goal

使用MIT-BIT的DATA SET，透過深度學習做出判斷心律不整的模型。



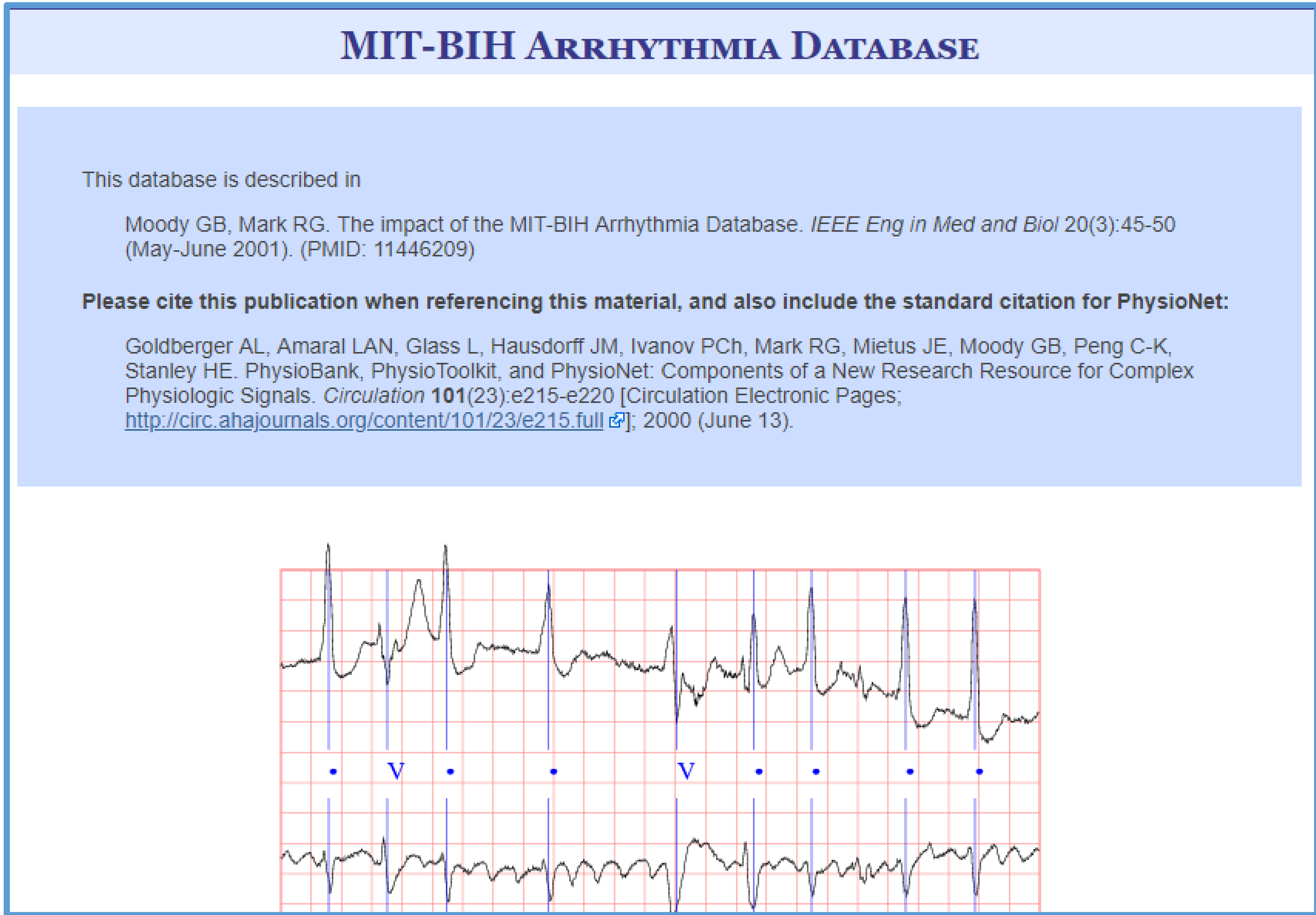
訓練與測試

Reference annotations	Signals	Header
100.atr	100.dat	100.hea
101.atr	101.dat	101.hea
102.atr	102.dat	102.hea
103.atr	103.dat	103.hea
104.atr	104.dat	104.hea
105.atr	105.dat	105.hea
106.atr	106.dat	106.hea
107.atr	107.dat	107.hea
108.atr	108.dat	108.hea
109.atr	109.dat	109.hea
111.atr	111.dat	111.hea

https://www.physionet.org/physiobank/database/mitdb/
總共48筆資料,取前38筆為訓練資料
後10筆為測試資料
每人於6/26號期末展示結果

MIT-BIH Dataset

美國麻省理工學院提供，研究心律失常的數據庫。目前國際上公認可作為標準的心電數據庫之一。



MIT-BIH Arrhythmia Database

This database is described in

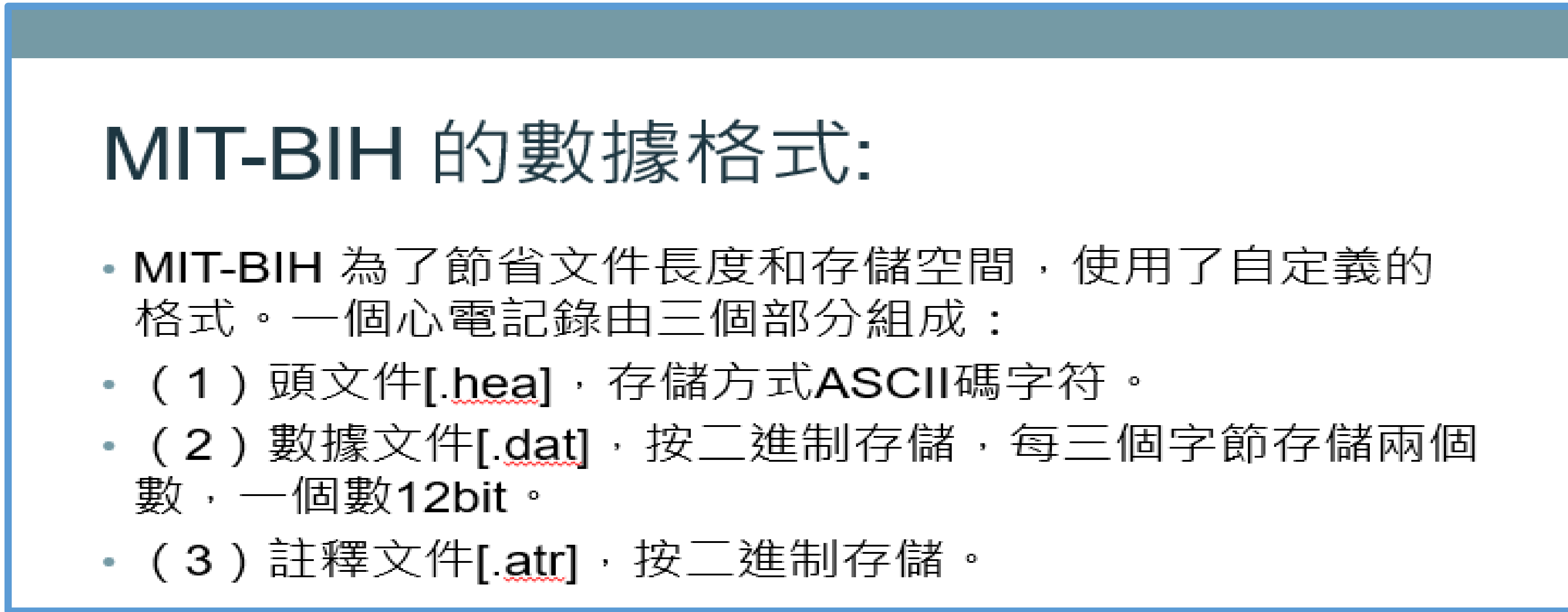
Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. *IEEE Eng in Med and Biol* 20(3):45-50 (May-June 2001). (PMID: 11446209)

Please cite this publication when referencing this material, and also include the standard citation for PhysioNet:

Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* **101**(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13).

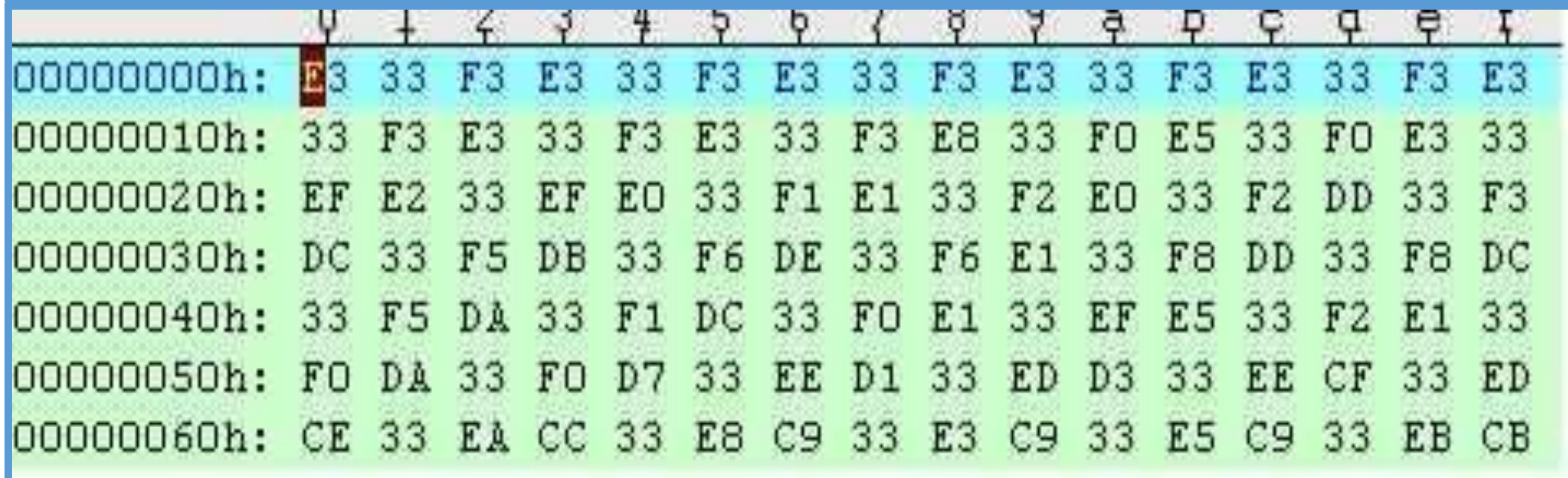
MIT-BIH Dataset

麻煩到令人想放棄的資料集



MIT-BIH 的數據格式:

- MIT-BIH 為了節省文件長度和存儲空間，使用了自定義的格式。一個心電記錄由三個部分組成：
- （1）頭文件[.hea]，存儲方式ASCII碼字符。
- （2）數據文件[.dat]，按二進制存儲，每三個字節存儲兩個數，一個數12bit。
- （3）註釋文件[.atr]，按二進制存儲。



```
             0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: E3 33 F3 E3 33 F3 E3 33 F3 E3 33 F3 E3 33 F3 E3
00000010h: 33 F3 E3 33 F3 E3 33 F3 E8 33 F0 E5 33 F0 E3 33
00000020h: EF E2 33 EF E0 33 F1 E1 33 F2 E0 33 F2 DD 33 F3
00000030h: DC 33 F5 DB 33 F6 DE 33 F6 E1 33 F8 DD 33 F8 DC
00000040h: 33 F5 DA 33 F1 DC 33 F0 E1 33 EF E5 33 F2 E1 33
00000050h: F0 DA 33 F0 D7 33 EE D1 33 ED D3 33 EE CF 33 ED
00000060h: CE 33 EA CC 33 E8 C9 33 E3 C9 33 E5 C9 33 EB CB
```

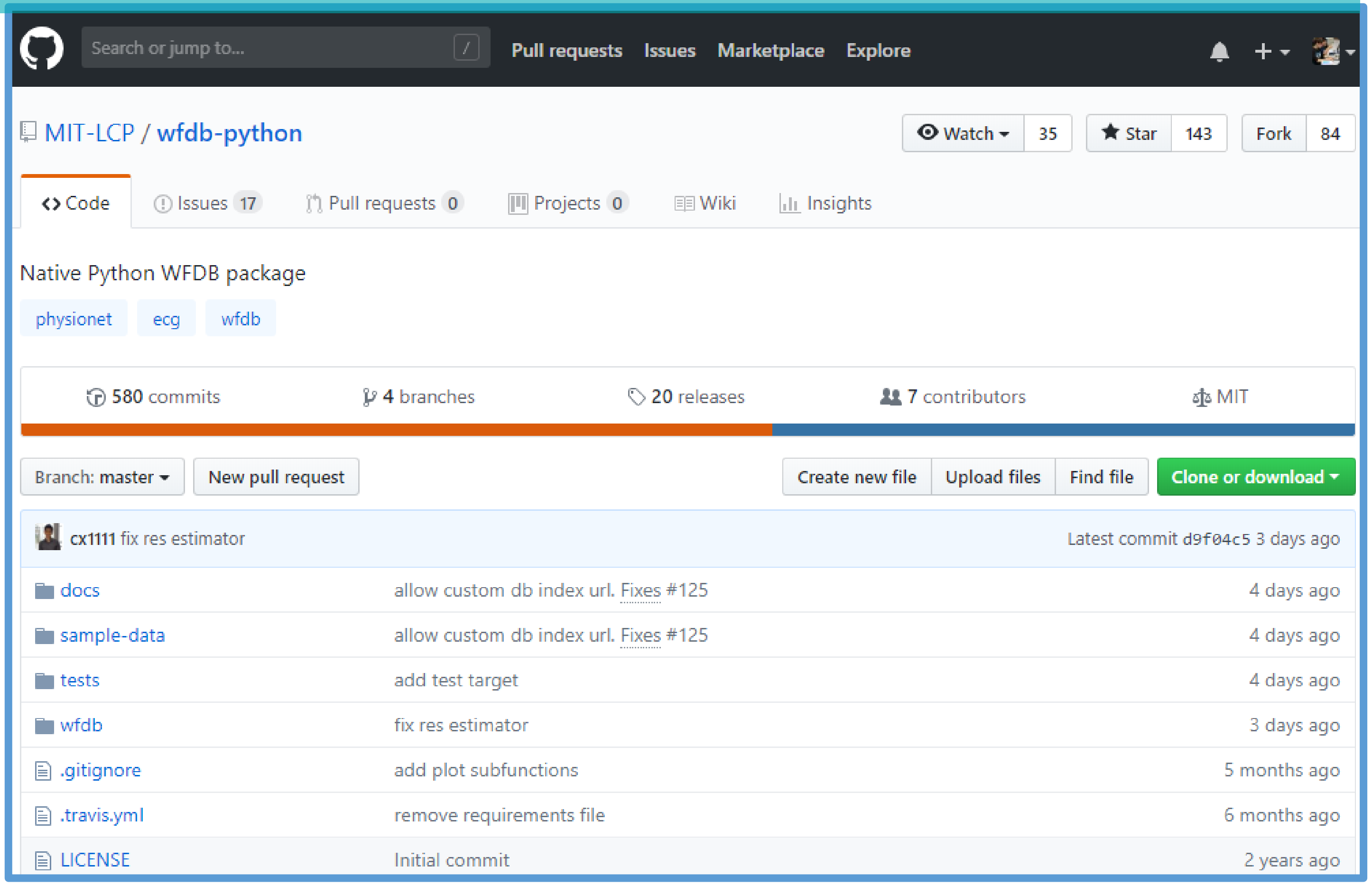
看到數據集後的反應

上帝為你關了一扇門

那是因為他要你吹冷氣

冷氣來啦! WFDB

waveform-database (WFDB) package



Search or jump to... / Pull requests Issues Marketplace Explore

MIT-LCP / **wfdb-python**

Watch 35 | Star 143 | Fork 84

Code | Issues 17 | Pull requests 0 | Projects 0 | Wiki | Insights

Native Python WFDB package

physionet ecg wfdb

580 commits | 4 branches | 20 releases | 7 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

cx1111 fix res estimator — Latest commit d9f04c5 3 days ago

docs	allow custom db index url. Fixes #125	4 days ago
sample-data	allow custom db index url. Fixes #125	4 days ago
tests	add test target	4 days ago
wfdb	fix res estimator	3 days ago
.gitignore	add plot subfunctions	5 months ago
.travis.yml	remove requirements file	6 months ago
LICENSE	Initial commit	2 years ago

Introduction

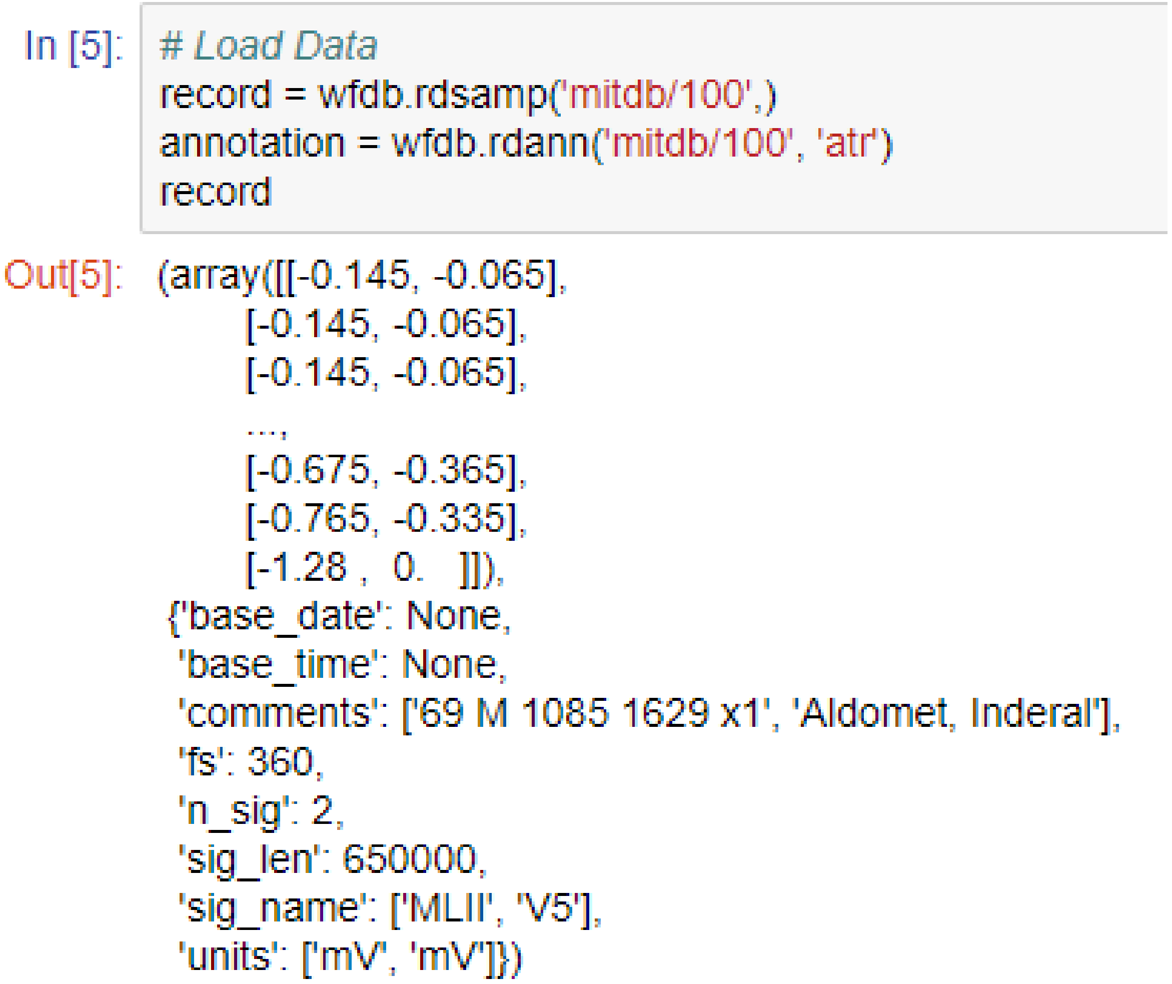
The native Python waveform-database (WFDB) package.

A library of tools for reading, writing, and processing WFDB signals and annotations.

Core components of this package are based on the original WFDB specifications. This package does not contain the exact same functionality as the original WFDB package. It aims to implement as many of its core features as possible, with user-friendly APIs.

Additional useful physiological signal-processing tools are added over time.

CODE片段



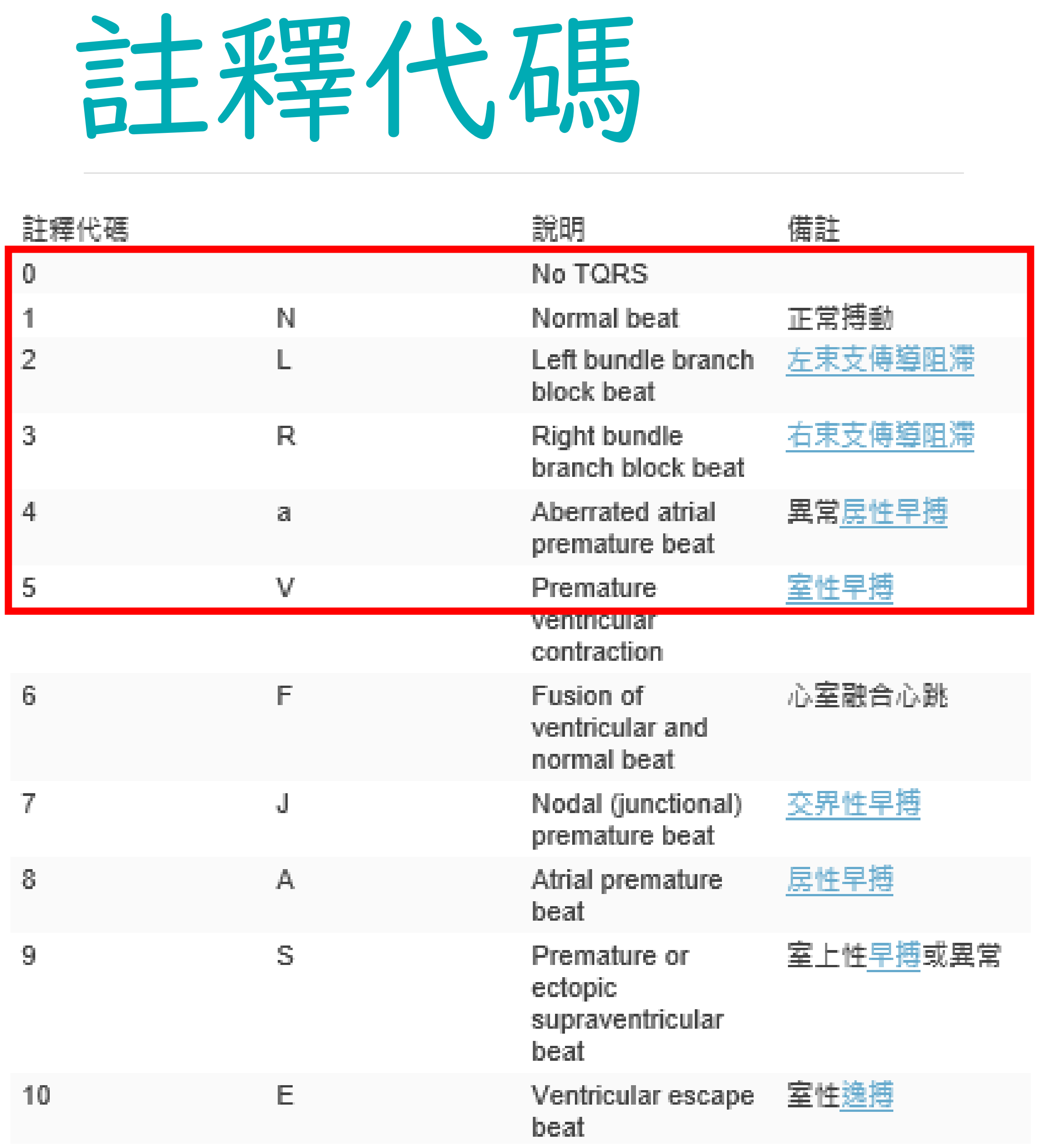
```
In [5]:  # Load Data
         record = wfdb.rdsamp('mitdb/100',)
         annotation = wfdb.rdann('mitdb/100', 'atr')
         record

Out[5]:  (array([[-0.145, -0.065],
                 [-0.145, -0.065],
                 [-0.145, -0.065],
                 ...,
                 [-0.675, -0.365],
                 [-0.765, -0.335],
                 [-1.28 ,  0.   ]]),
          {'base_date': None,
           'base_time': None,
           'comments': ['69 M 1085 1629 x1', 'Aldomet, Inderal'],
           'fs': 360,
           'n_sig': 2,
           'sig_len': 650000,
           'sig_name': ['MLII', 'V5'],
           'units': ['mV', 'mV']})
```

Introduction to Data Structure of WFDB

- record: 記錄.dat檔案中的數據，每筆.dat檔案有兩個探測點(各蒐集65萬筆資料)

- annotation: 紀錄.atr檔案中的資料，該筆資料包含了兩個部分:
  1.採樣點位置
  2.該位置的症狀判斷

Introduction to Data Structure of WFDB



註釋代碼

註釋代碼		說明	備註
0		No TQRS	
1	N	Normal beat	正常搏動
2	L	Left bundle branch block beat	左束支傳導阻滯
3	R	Right bundle branch block beat	右束支傳導阻滯
4	a	Aberrated atrial premature beat	異常房性早搏
5	V	Premature ventricular contraction	室性早搏
6	F	Fusion of ventricular and normal beat	心室融合心跳
7	J	Nodal (junctional) premature beat	交界性早搏
8	A	Atrial premature beat	房性早搏
9	S	Premature or ectopic supraventricular beat	室上性早搏或異常
10	E	Ventricular escape beat	室性逸搏

- record: 記錄.dat檔案中的數據,每筆.dat檔案有兩個探測點(各蒐集65萬筆資料)

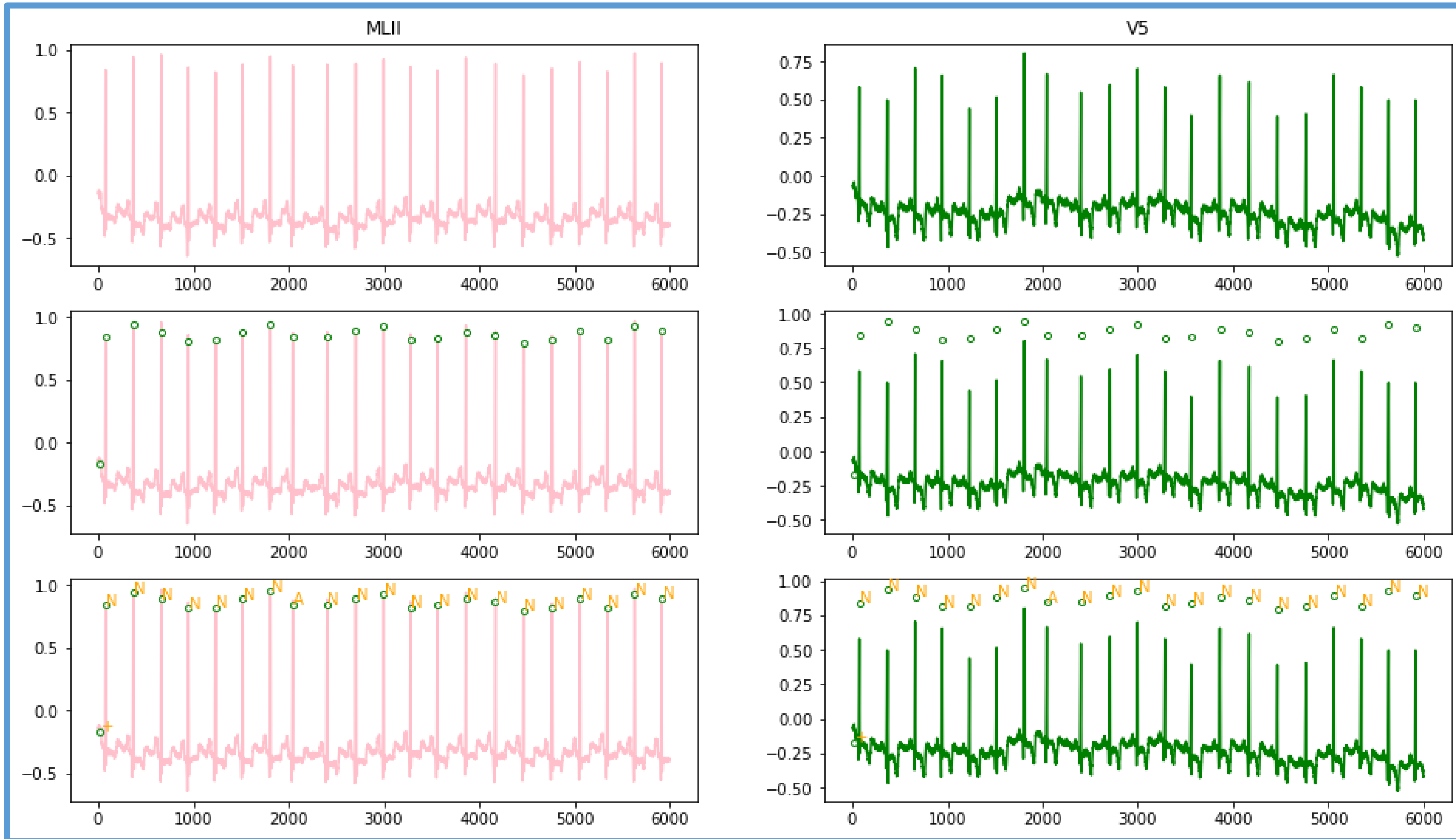
- annotation: 紀錄.atr檔案中的資料,該筆資料包含了兩個部分:
  1.採樣點位置
  2.該位置的症狀判斷

PART TWO

資料分析

Data Visualization

透過資料視覺化
觀察數據與標記的關係

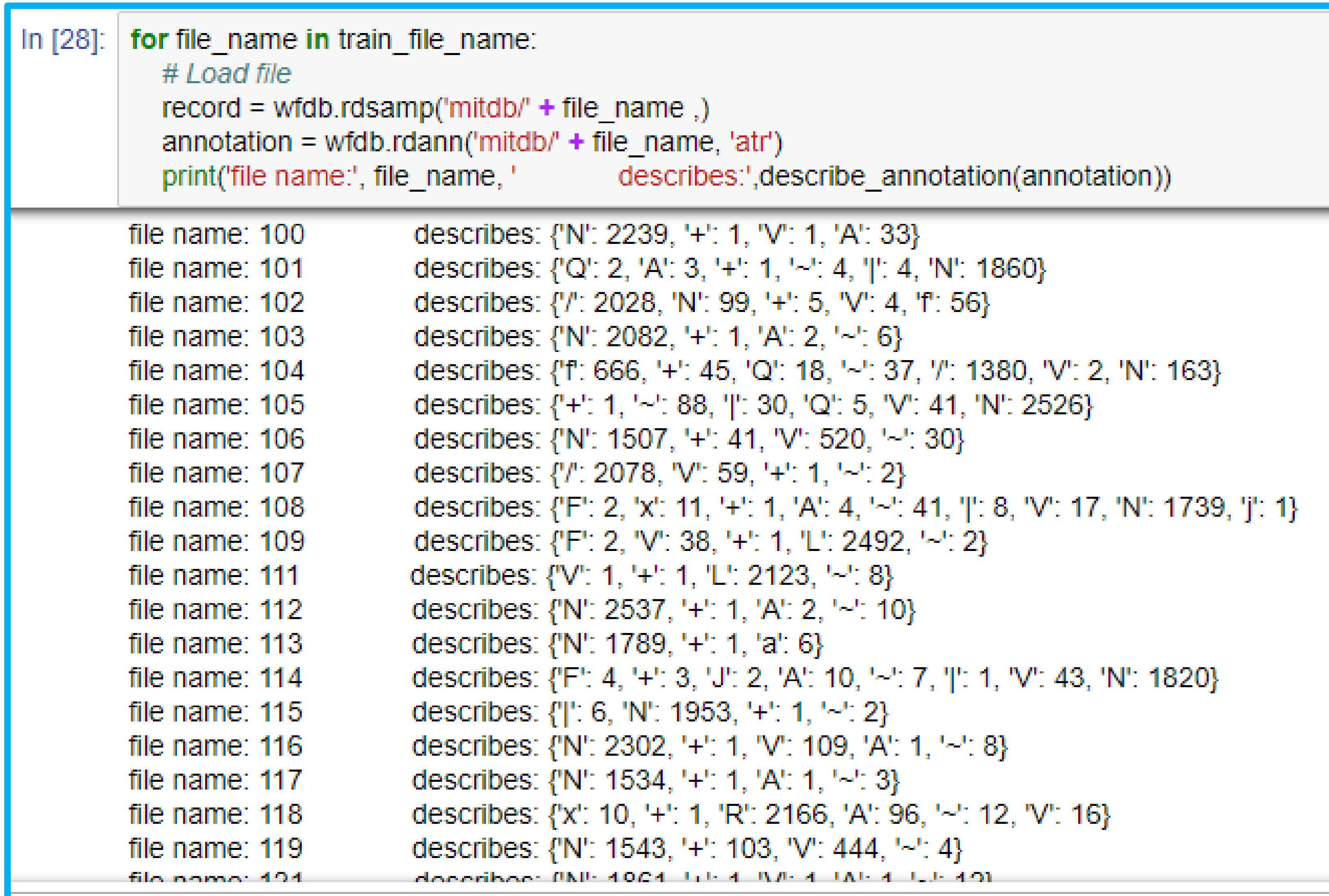


Label

statistical analysis

透過資料視覺化
觀察數據與標記的關係

We need: N、L、R、a、V



```
In [28]: for file_name in train_file_name:
             # Load file
             record = wfdb.rdsamp('mitdb/' + file_name ,)
             annotation = wfdb.rdann('mitdb/' + file_name, 'atr')
             print('file name:', file_name, '          describes:',describe_annotation(annotation))
```

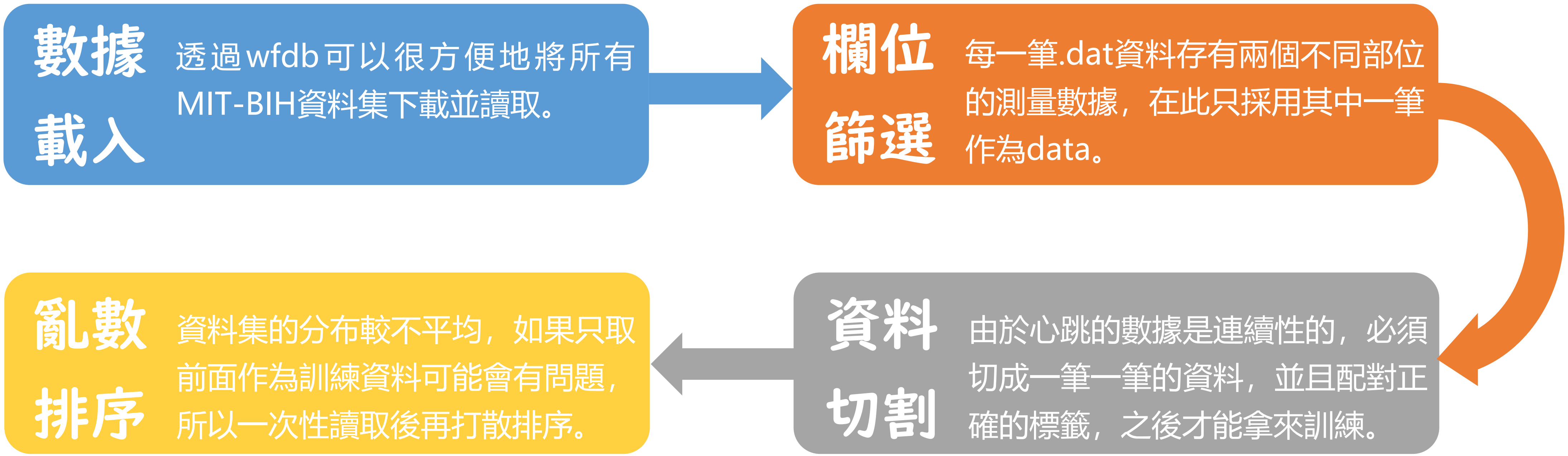
```
         file name: 100          describes: {'N': 2239, '+': 1, 'V': 1, 'A': 33}
         file name: 101          describes: {'Q': 2, 'A': 3, '+': 1, '~': 4, '|': 4, 'N': 1860}
         file name: 102          describes: {'/': 2028, 'N': 99, '+': 5, 'V': 4, 'f': 56}
         file name: 103          describes: {'N': 2082, '+': 1, 'A': 2, '~': 6}
         file name: 104          describes: {'f': 666, '+': 45, 'Q': 18, '~': 37, '/': 1380, 'V': 2, 'N': 163}
         file name: 105          describes: {'+': 1, '~': 88, '|': 30, 'Q': 5, 'V': 41, 'N': 2526}
         file name: 106          describes: {'N': 1507, '+': 41, 'V': 520, '~': 30}
         file name: 107          describes: {'/': 2078, 'V': 59, '+': 1, '~': 2}
         file name: 108          describes: {'F': 2, 'x': 11, '+': 1, 'A': 4, '~': 41, '|': 8, 'V': 17, 'N': 1739, 'j': 1}
         file name: 109          describes: {'F': 2, 'V': 38, '+': 1, 'L': 2492, '~': 2}
         file name: 111          describes: {'V': 1, '+': 1, 'L': 2123, '~': 8}
         file name: 112          describes: {'N': 2537, '+': 1, 'A': 2, '~': 10}
         file name: 113          describes: {'N': 1789, '+': 1, 'a': 6}
         file name: 114          describes: {'F': 4, '+': 3, 'J': 2, 'A': 10, '~': 7, '|': 1, 'V': 43, 'N': 1820}
         file name: 115          describes: {'|': 6, 'N': 1953, '+': 1, '~': 2}
         file name: 116          describes: {'N': 2302, '+': 1, 'V': 109, 'A': 1, '~': 8}
         file name: 117          describes: {'N': 1534, '+': 1, 'A': 1, '~': 3}
         file name: 118          describes: {'x': 10, '+': 1, 'R': 2166, 'A': 96, '~': 12, 'V': 16}
         file name: 119          describes: {'N': 1543, '+': 103, 'V': 444, '~': 4}
         file name: 121          describes: {'N': 1861, '+': 1, 'V': 1, 'A': 1, '~': 12}
```

PART THREE

程式設計

Data Preprocessing

先對資料作一些初步的處理



數據載入

透過wfdb可以很方便地將所有MIT-BIH資料集下載並讀取。

欄位篩選

每一筆.dat資料存有兩個不同部位的測量數據，在此只採用其中一筆作為data。

資料切割

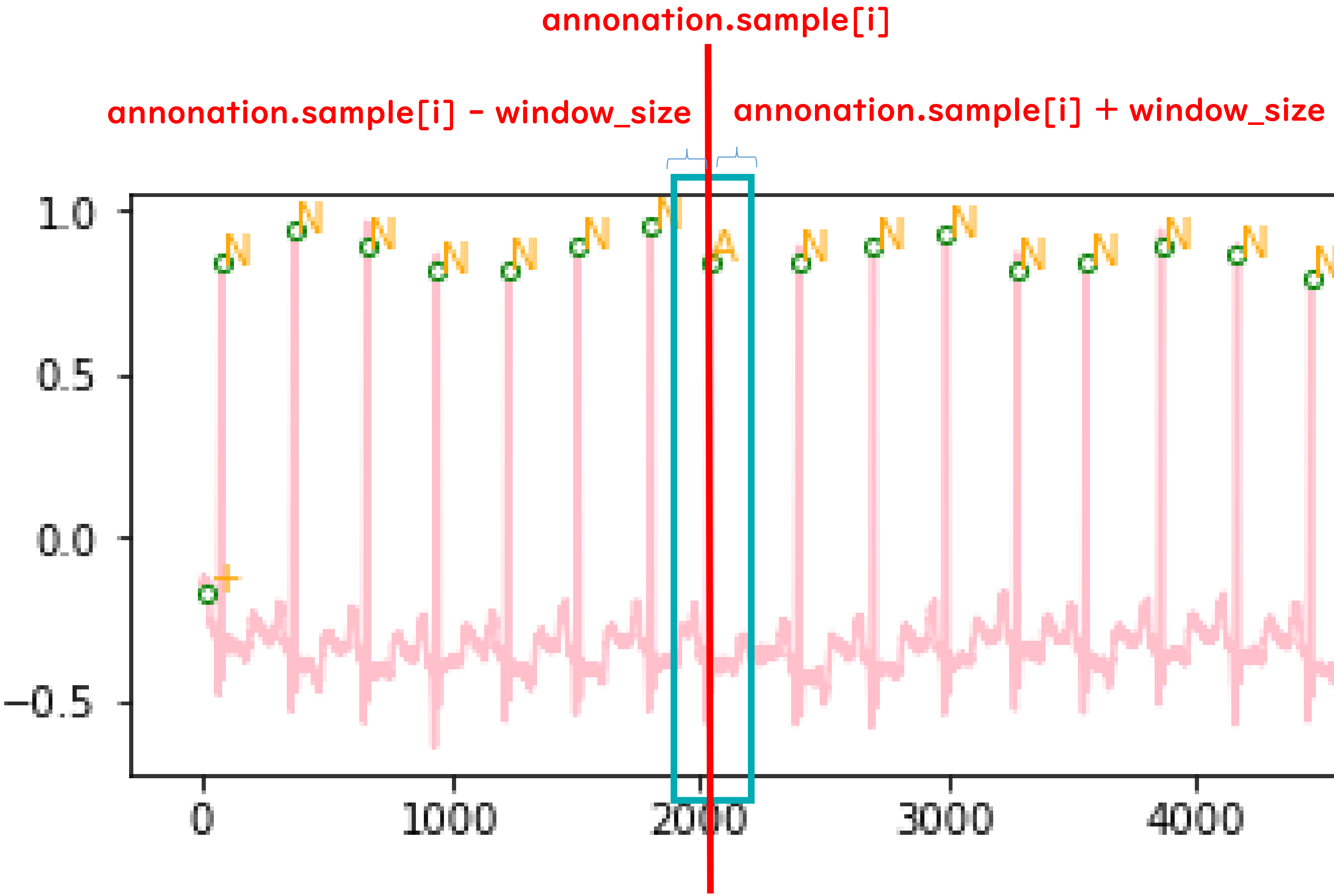
由於心跳的數據是連續性的，必須切成一筆一筆的資料，並且配對正確的標籤，之後才能拿來訓練。

亂數排序

資料集的分布較不平均，如果只取前面作為訓練資料可能會有問題，所以一次性讀取後再打散排序。

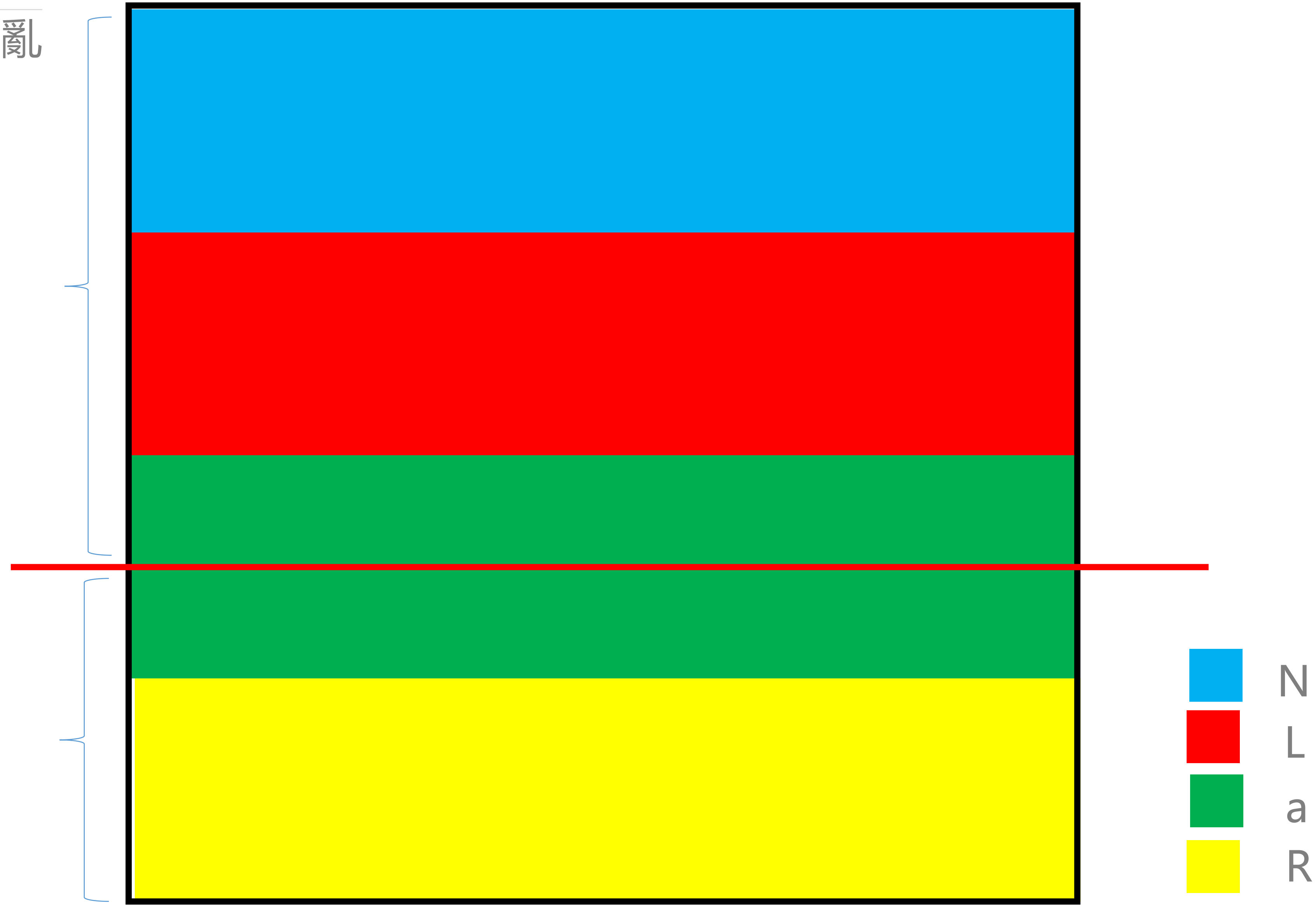
資料切割

將連續的心跳數據離散化



亂數排序

使得資料的分布較為雜亂



training set

testing set

🟦 N

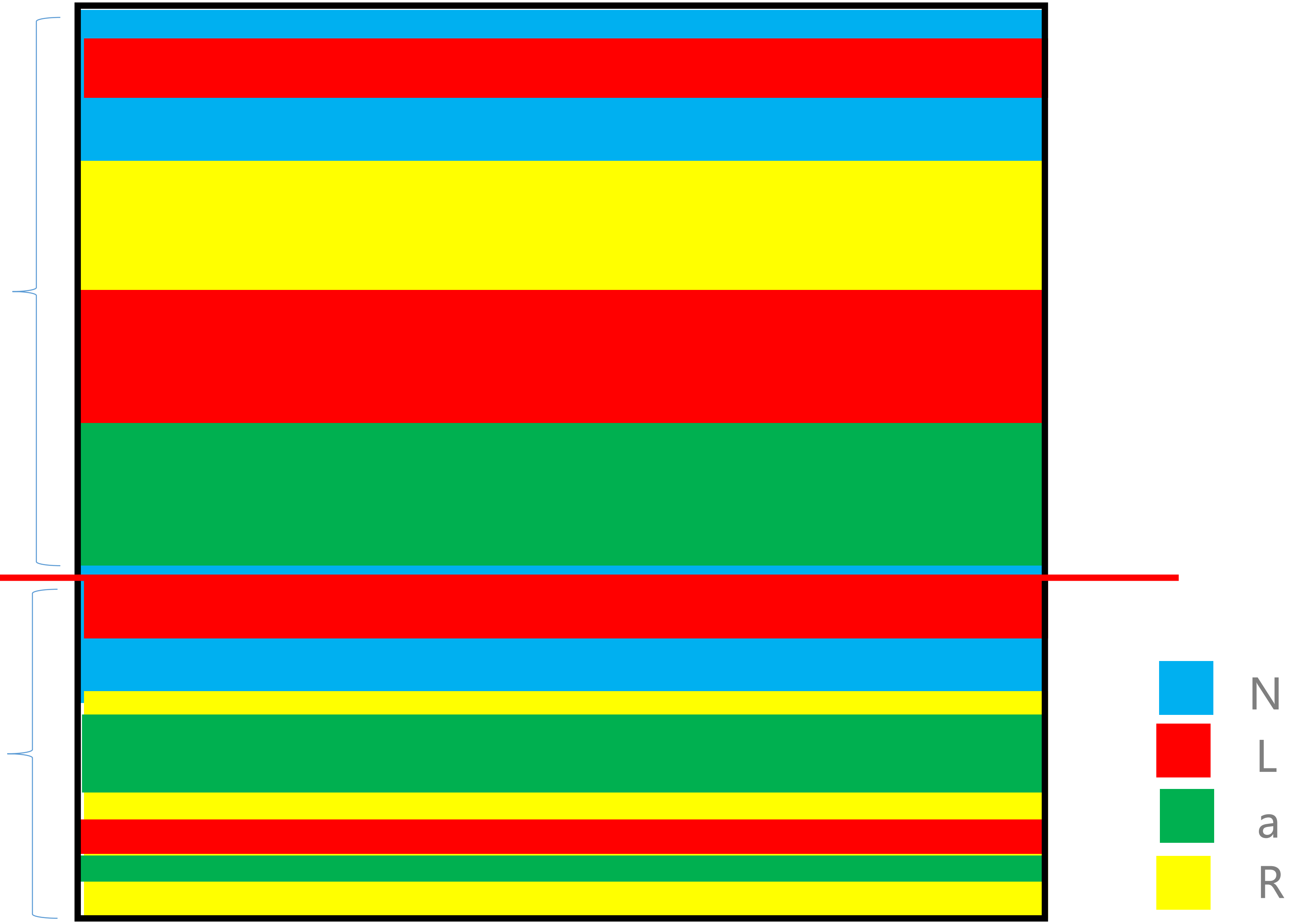
🟥 L

🟩 a

🟨 R

亂數排序

使得資料的分布較為雜亂



training set

testing set

🟦 N

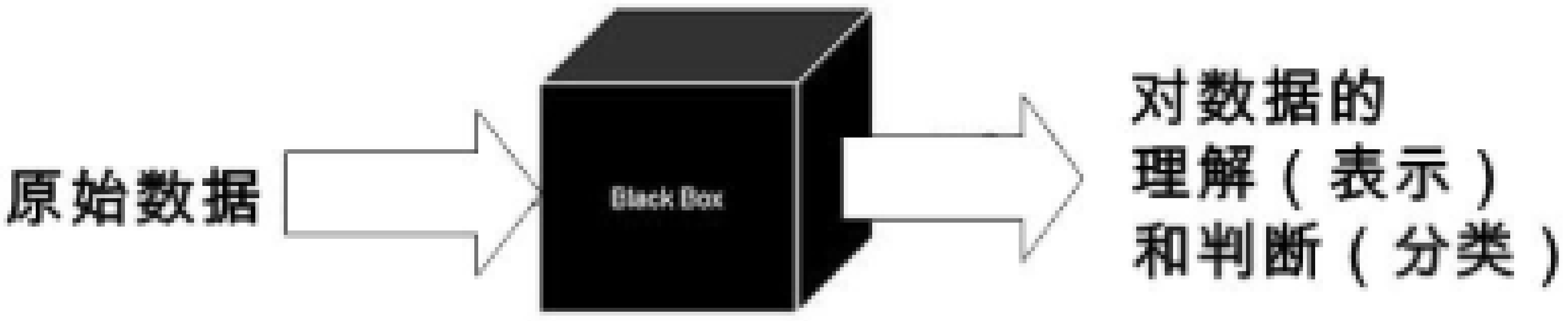
🟥 L

🟩 a

🟨 R

LSTM Model

機器學習就像是黑盒子一樣..



PART FOUR

結論

先說結論

In conclusion

Training Loss

0.1009

with the best result

Test Accurancy

98%

with the best result

```
LR = 0.01
ITERATIONS = 100000
NUM_UNITS = 100
```

```
Model: BasicLSTMCell
loss function:
softmax_cross_entropy
Optimizer :AdadeltaOptimizer
```

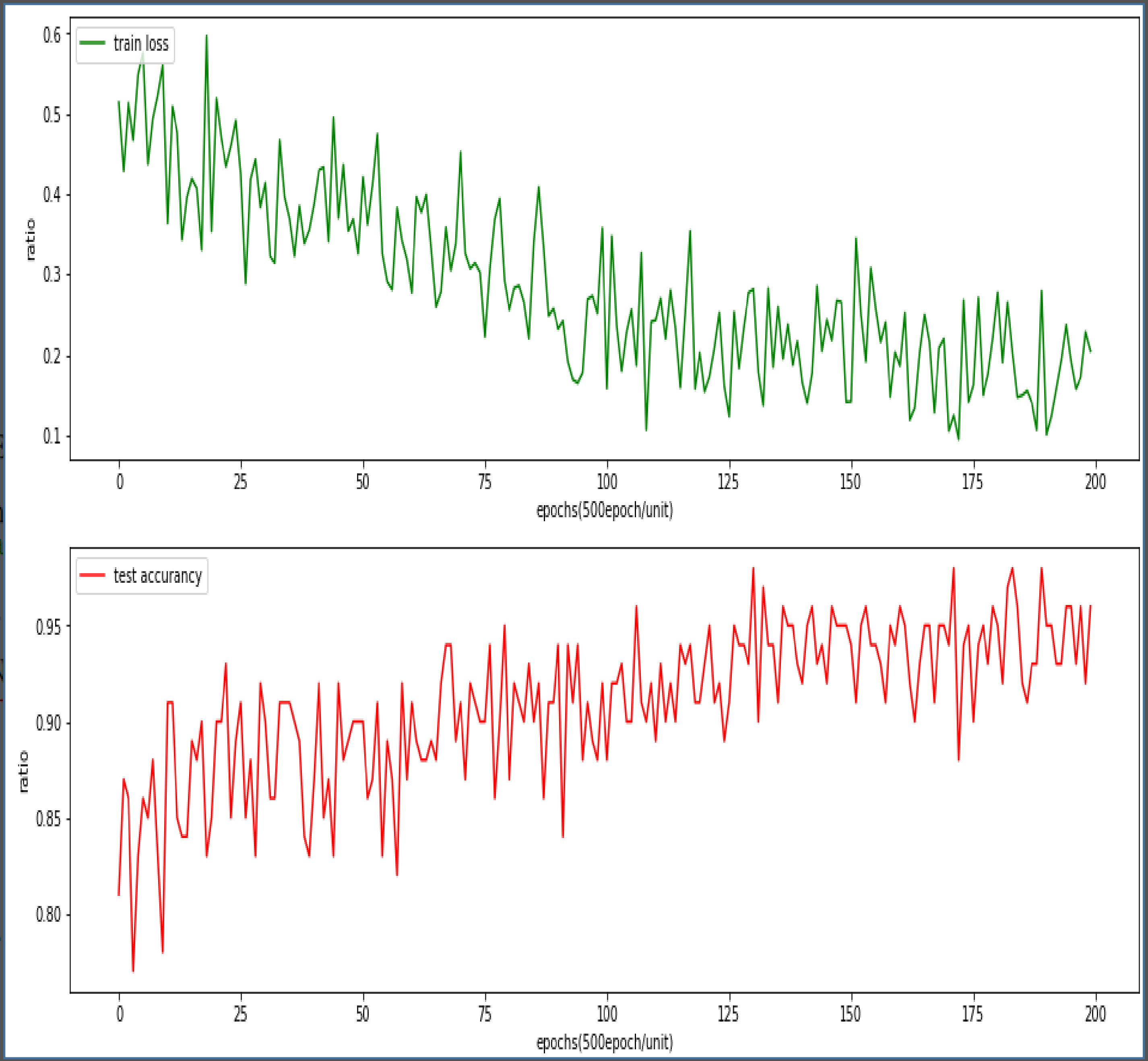
```
BATCH_SIZE = 100
TIME_STEP = 10
INPUT_SIZE = 20
```

Result of LSTM Training

透過系上提供的GPU工作站進行Data Training，由於tf自帶Optimizer，在learning rate上不用那麼著墨，比較注重在訓練次數的調整上。

在Epochs數量少時正確率還不是那麼明顯，不過把Epochs大幅提高(從8000->100000次)後，有了明顯的提升。

跑了大約1個小時(順便看世足)，最後得到了平均95%以上的Accurancy。

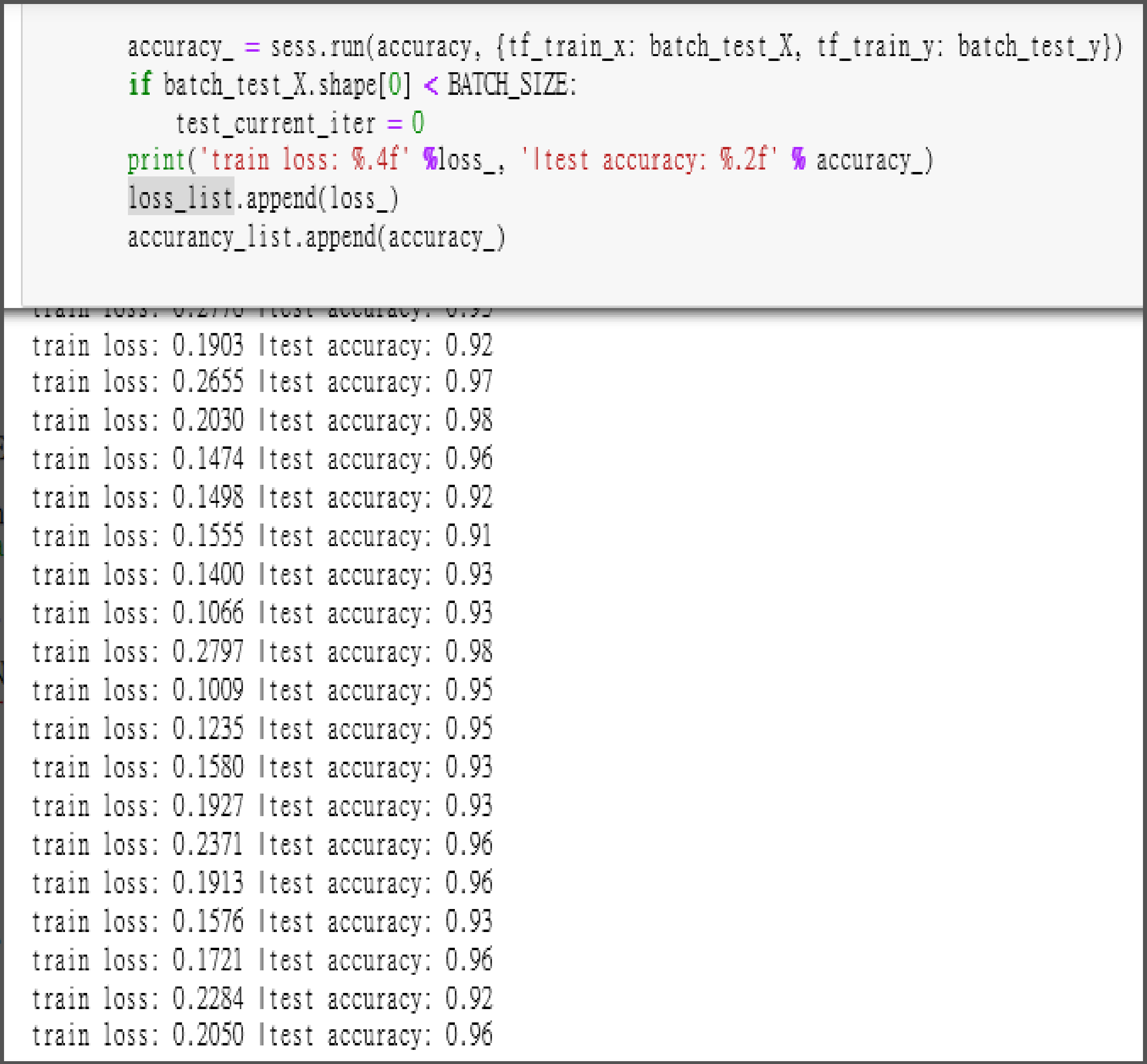


Result of LSTM Training

透過系上提供的GPU工作站進行Data Training，由於tf自帶Optimizer，在learning rate上不用那麼著墨，比較注重在訓練次數的調整上。

在Epochs數量少時正確率還不是那麼明顯，不過把Epochs大幅提高(從8000->100000次)後，有了明顯的提升。

跑了大約1個小時(順便看世足)，最後得到了平均95%以上的Accurancy。



```
accuracy_ = sess.run(accuracy, {tf_train_x: batch_test_X, tf_train_y: batch_test_y})
if batch_test_X.shape[0] < BATCH_SIZE:
    test_current_iter = 0
print('train loss: %.4f' %loss_, '|test accuracy: %.2f' % accuracy_)
loss_list.append(loss_)
accurancy_list.append(accuracy_)
```

```
train loss: 0.1903 |test accuracy: 0.92
train loss: 0.2655 |test accuracy: 0.97
train loss: 0.2030 |test accuracy: 0.98
train loss: 0.1474 |test accuracy: 0.96
train loss: 0.1498 |test accuracy: 0.92
train loss: 0.1555 |test accuracy: 0.91
train loss: 0.1400 |test accuracy: 0.93
train loss: 0.1066 |test accuracy: 0.93
train loss: 0.2797 |test accuracy: 0.98
train loss: 0.1009 |test accuracy: 0.95
train loss: 0.1235 |test accuracy: 0.95
train loss: 0.1580 |test accuracy: 0.93
train loss: 0.1927 |test accuracy: 0.93
train loss: 0.2371 |test accuracy: 0.96
train loss: 0.1913 |test accuracy: 0.96
train loss: 0.1576 |test accuracy: 0.93
train loss: 0.1721 |test accuracy: 0.96
train loss: 0.2284 |test accuracy: 0.92
train loss: 0.2050 |test accuracy: 0.96
```

特別感謝

宗鴻跟淙垣在技術上的諸多指導