

期末專題

Automatic Sleep Stage Classification

107 學年度第一學期 台大電機生醫工程實驗

第三組

解正平 B04901020

張景程 B04901138

劉維凱 B04901153

指導助教：趙珮妤 實驗室：明達館304室

目次

一、 Abstract	p.3
二、 Introduction	p.3
三、 Data Acquisition	p.4
四、 Data Preprocessing	p.6
五、 Classification Methods	p.7
六、 Traditional Learning	p.8
七、 Deep Learning	p.13
八、 Results	p.17
九、 Discussion	p.22
十、 Application - alarm clock	p.23
十一、 Conclusion	p.28
十二、 Reference	p.28

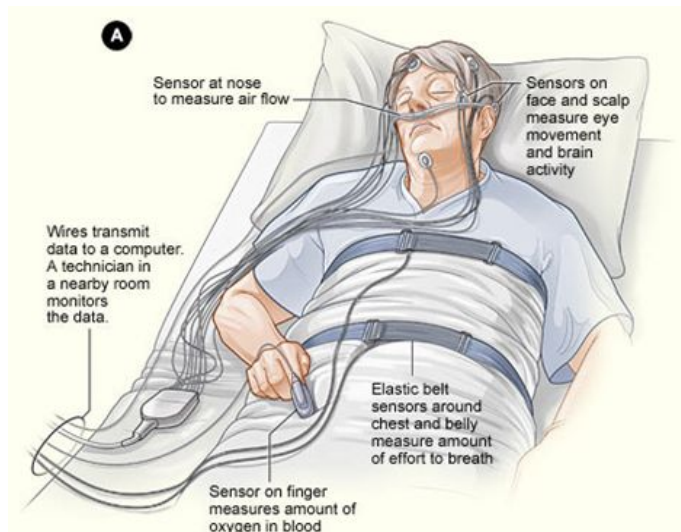
一、Abstract

使用MindWave mobile量測人體EEG生理訊號，並透過XGBoost 以及deep neural network兩種machine learning方法來達到automatic sleep stage classification，並實作手機鬧鐘app來應用模型，即時紀錄並判斷睡眠狀態，並在適合起床的階段響鈴起床。

二、Introduction

睡眠是我們生活中很重要的部分，尤其現今人們因為健康因素越來越重視睡眠品質，比如說整個晚上常睡不著或是無法熟睡都是很嚴重的問題，市面上有很多穿戴式裝置目的在監測睡眠品質來發現問題，透過環境聲音分析，或是心跳速率等方式判斷睡眠狀態。本次專題希望利用腦波EEG的量測來達到automatic sleep stage classification (ASSC)，不同以往使用其他方式來偵測睡眠，更改善以前都必須由專業醫師診斷腦波，自動偵測可以即時快速、精準地記錄睡眠變化，以幫助診斷睡眠窒息症、失眠、嗜睡症等健康問題。

Rechtschaffen and Kales是判斷睡眠階段的標準方法，階段可以分為wakefulness (W)，non-rapid eye movement sleep stage (NREM) 還有rapid eye movement sleep stage (REM)，其中NREM還可以再分為4個stage (s1-s4)，而s3及s4又可以併稱slow wave sleep (SWS)。睡眠週期多為以上幾個階段循環(見下圖)，時間約為90分鐘，為了能夠成功判斷這些睡眠狀態，最好的檢驗方式多是透過睡眠多項生理檢查polysomnography (PSG)，其中會量測腦電圖EEG、眼電圖EOG、心電圖ECG、肌電圖EMG等訊號，分析這些訊號的特性就可以有效診斷人的睡眠問題，本次專題的訊號分析重點是腦電圖EEG，利用腦波在不同階段的差異來判斷睡眠階段。



清醒(w)的時候腦波多屬於beta波，腦電壓較小也很隨機沒有甚麼規則，當進入睡眠第一階段(s1)，前半段剛閉眼多為明顯的alpha波，之後便是頻率較小的theta波，這階段多會出現入睡抽動(hypnic jerks)的現象。再來第二階段(s2)嗜睡眠佔最大部份的狀態，腦波主要都是呈現theta波，這個階段會出現突然強烈的紡錘波(spindles waves)以及震幅極大的k-complexes，而且較少眼動現象，人體心跳變慢溫度漸低。第三階段(s3)腦波開始出現頻率小的delta波，第四階段(s4)則是幾乎都是更小頻率的delta波，通常這兩個階段會一起判斷為深層睡眠，夢遊或是說夢話都是出現在這個時候。睡眠cycle的最後是快速眼動期(REM)，人體心跳會逐漸加快，腦波訊號電壓漸小，頻率也變得較高較不規則，我們通常作夢都是發生在這個階段，這個階段結束會再進入s1-s4-REM的循環，而循環多次會REM的比例會越來越高，深層睡眠的比例會漸少。以上幾個階段幾乎都有明顯得特徵，透過不同頻段的訊號分析可以清楚判斷，然而最大的困難是s1與REM的區分，因為兩者都是電壓較小頻率較高的狀態，以前實驗多是以眼電圖EOG判斷是否有快速眼動當作依據，本專題因為只使用EEG因此會是需要克服的問題。

本次專題目標是機器自動判斷人類睡眠階段，並以此模型設計應用來改善我們的生活。平常我們設定鬧鐘都是以時間當作依據，並未考慮睡眠的狀態，而使得鬧鐘打斷我們睡眠並影響品質，我們希望可以設計一款手機app，藉由穿戴式裝置量測腦電 EEG 訊號，自動判斷睡眠階段，並在使用者設定的時間區間內，以睡眠階段當作依據在適合的時候響鈴，期望可以改善我們起床的心情與狀態，以達到睡眠品質最佳。

三、Data Acquisition

1. [Sleep Recordings and Hypnograms in European Data Format]

EDF [Expanded] database (<https://physionet.org/physiobank/database/sleep-edfx/>) 在ASSC領域許多人都使用Physionet SleepEDF database (如上連結)，資料共分成兩種file，分別是SC file (Sleep Cassette)及ST file (Sleep Telemetry)，資料的sampling rate = 100 Hz，以30秒為單位當作一個epoch，每個epoch有3000點及由專業醫生判斷的一個label。其中實驗量測的EEG channel有兩種Fpz-Cz和Pz-Oz，本實驗使用Fpz-Cz，因為以前paper的結果多以此channel的performance最佳。

- [SC*]

20位受試者，每位共測量兩次，每次20小時左右，受試者在家測量。

由於這份檔案的受試者多處於wake狀態，因此需要先將data取出真的有在睡眠的時間，我們以第一次非wake當狀態開始，當連續五次wake且epoch有超過1000當作狀態結束。

- [ST*]

22位受試者，每位共測量一次，每次9小時左右，受試者在醫院測量。

由於這份檔案的受試者多是有失眠的問題，非正常一般人的腦波訊號，因此與SC*的file不屬相同類型。我們將每個受試者訊號前5分鐘及最後2分半去除，也就是前10個epoch及後5個epoch。

兩份資料總共有62份睡眠紀錄，其中40份是一般測資，22份是受試者有失眠狀況的測資，為配合本專題已判斷睡眠階段為主，我們只以40份一般測資當作我們使用的資料。下表是我們統計的結果。

epochs	total	W	S1	S2	S3	R
All files	62255	9643 (15.4%)	4809 (7.7%)	27187 (43.6%)	8805 (14.1%)	11811 (18.9%)
SC* files	41599	7576 (18.2%)	2804 (6.7%)	17799 (42.8%)	5703 (13.7%)	7717 (18.6%)
ST* files	20656	2067 (10.0%)	2005 (9.7%)	9388 (45.4%)	3102 (15.0%)	4094 (19.8%)

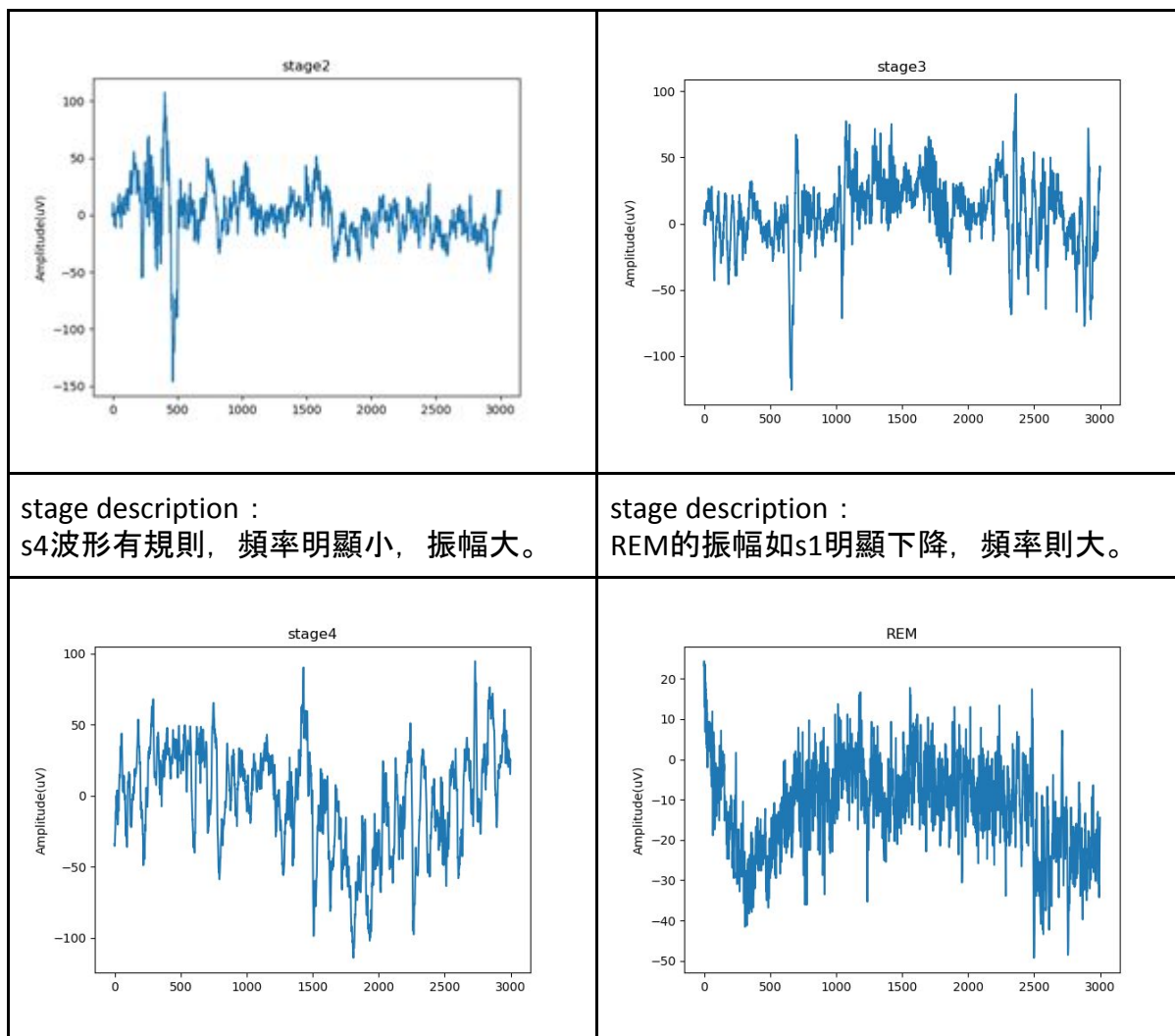
2. [EDF browser]

為了將網站提供的[*-PSG.edf]及[*-Hypnogram.edf]轉成我們可以使用的[.csv]檔，我們使用EDF browser將波形視覺化並且輸出每個amplitude的電壓。其中視覺化之後可以觀察發現前後epoch常常有許多wake影響label而使data分佈不平衡，可能導致model很難建立，因此必須特別處理，減少wake的數據。



3. [data visuliation]

<p>stage description : wake波形不規則， 振幅較大。</p>	<p>stage description : s1振幅明顯下降。</p>
<p>The plot titled 'Wake' shows EEG amplitude in microvolts (uV) on the y-axis (ranging from -150 to 150) against time on the x-axis (ranging from 0 to 3000). The waveform is highly irregular and noisy, with large positive and negative excursions, indicating a state of wakefulness.</p>	<p>The plot titled 'stage1' shows EEG amplitude in microvolts (uV) on the y-axis (ranging from -20 to 40) against time on the x-axis (ranging from 0 to 3000). The amplitude is significantly lower and less variable than in the 'Wake' stage, indicating a transition to a lighter sleep stage.</p>
<p>stage description : s2可以發現k-complex波。</p>	<p>stage description : s3頻率較小， 仍有k-complex， 振幅大。</p>



四、Data Preprocessing

根據本專題不同的machine learning方式，我們會做兩種不同的data preprocessing。

1. traditional traing methods :

因為腦波EEG的data變化很大，常常量測也會有雜訊，因此我們使用4階 Butterworth band pass filter，passband設為0.1Hz到45Hz，目的是來濾掉高頻訊號。接下來處理極端值的問題，我們把具有振幅大小超過400uV的epoch捨棄掉，因為人體生理訊號並不會這麼高，甚至比k-complex還有劇烈，再來data我們有分是否有做過normalize，我們標準化的形式是對每個受試者整晚睡覺的訊號進行處理，以期能夠消除受試者之間的差異，最後我們將所有受試者(SC*及ST*)的epoch弄成一份檔案，期望傳統方法可以成功判斷睡眠階段。

2. deep neural network :

深層神經網路不太需要將data進行preprocessing，因為整個model會去學如何忽略雜訊，並且選擇重要的資訊來判斷，如果是極端值或是高頻的訊號，model會學到自己的filter將與其他數值相差太多的去除掉。然而model會有問題是使用的data相關性要高，才不會學到不正確的資訊，比如說我們最後決定只以受試者(SC*)當作training的檔案，沒有加入(*ST)是因為他們是不同族群的受試者。

五、Classification Method

Machine learning的方法有很多種，本專題的目標是對sleep stage做classification，我們使用傳統features的learning方法，以及近幾年流行的neural network，兩種方法步驟差異很多，大致有以下的流程：

1. Traditional Learning

傳統的方法必須先從原始訊號取出需要的feature，並從這些特徵當作training的參數。feature多寡以及可信度都對結果有巨大的影響，因此需要對特徵做篩選。

- feature extraction

根據對data的處理，可以抽出需要的特定特徵，比如說生理訊號有time或frequency的分析；預測年收入有年齡職業教育程度的參數；PM2.5有物質濃度濕度雨量等相關影響原因。feature extraction會直接改變你training方向，因為我們僅能從我們抽取出來的feature變化。

- feature selection

抽取出來的feature不一定每個對training都有幫助，因此有很多方法可以實作來選擇較重要的特徵，或是說達到dimension reduction，使用何種方法要依data性質而有不同，比如說圖片常使用PCA；binary classification常使用fisher score；其他像是mRMR、Sequential methods和RFE也都是常見的方法。

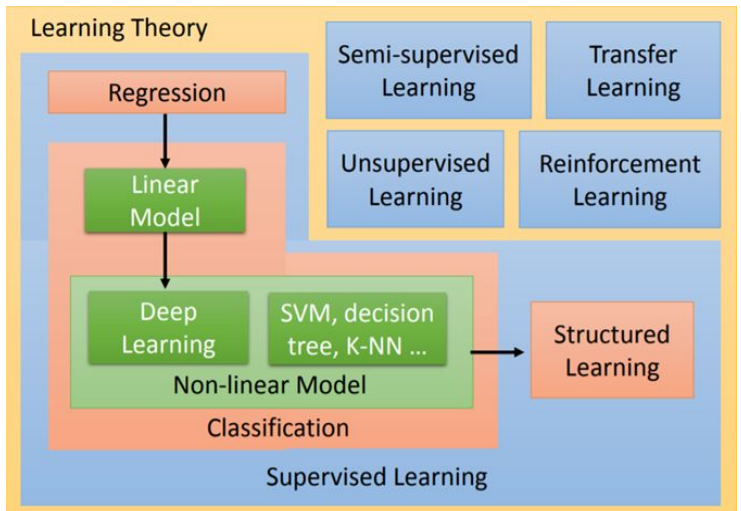
- classification method

在做classification的時候，目標是希望將data可以根據條件劃分不同label的區域，並且根據data之間的關係來判斷未知data的label。常見的方法有SVM(support vector machine)、KNN(knearest neighbor)、RF(randomforest)、DT(decision tree)。每種方法都差異很大，必須根據data的特性及分布使用。

2. Deep Learning

Deep learning是maching learning的其中一條分支，人們從大腦生物學中，神經元互相連接的模式取得靈感，而有了Neural Network的誕生，其目的是希望透過多層layer中的linear and non-linear transform特性，自動從資料中抽取出足以代表資料特性的feature。在傳統的機器學習中，feature通常是藉由人們設計演算法產生出來的，必須經過各領域的專家對資料進行許多的分析及研究，了解資料的特性後，才能產生出有用且效果良好的特徵，這樣的過程就是Feature engineering。

Deep learning具有自動達成feature extraction的能力，可以節省專家的人工抽取feature所花費的時間，並在Computer Vison、Natural Language Processing等高複雜度的task上達到極為驚人的優異表現，近年來也逐漸應用在生醫工程領域之中。



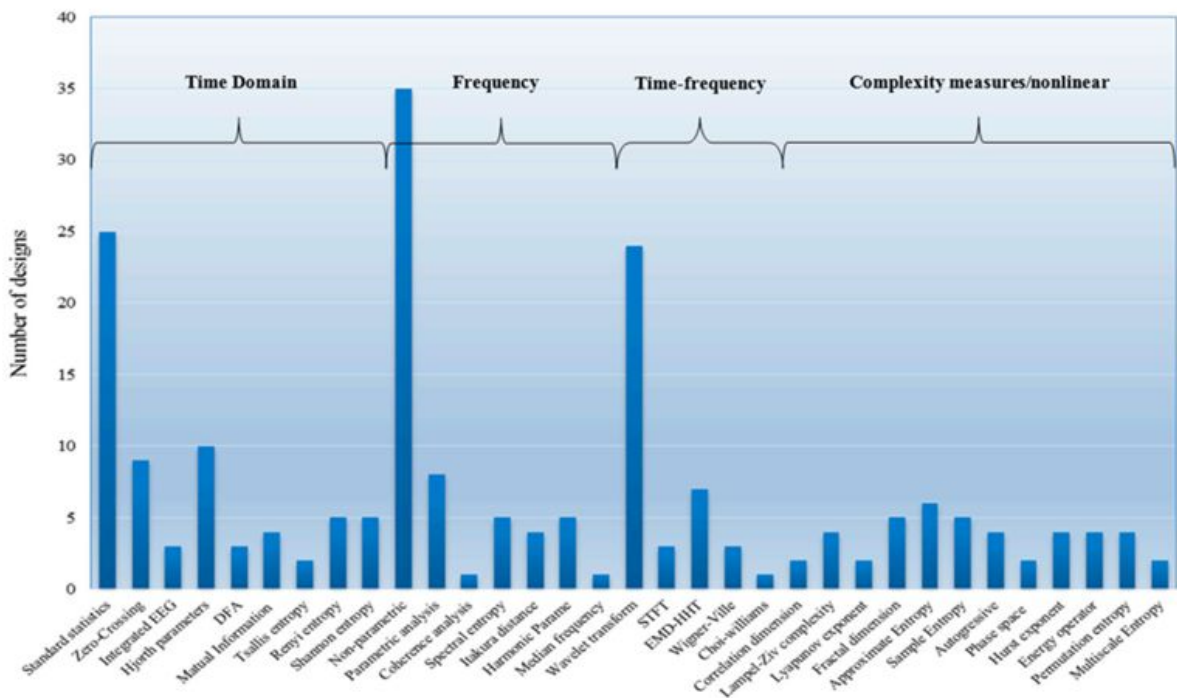
六、Traditional Learning

本專題共取出99個feature，並使用XGboost的方式做classification。

1. feature extraction

根據生理訊號EEG的分析，每個epoch的3000點discrete signal共抽出99種特徵，包含time analysis、frequency analysis、time frequency、complexity methods。

附圖是過去paper統計的使用方法，其中以time domain standard analysis、frequency domain non-parametric及time-frequency wavelet transform三種feature最多研究使用。



a. time analysis(15)

i. standard statistics : (8)

[mean、standard、skewness、kurtosis、first difference、normalize first difference、second difference、normalize second difference]
這些feature主要是分析EEG訊號在每個epoch的 central tendency, degree of dispersion, asymmetry, data peaks, troughs and flatness respectivel.

ii. zero crossing rate : (1)

計算波形隨著時間變化時，電壓amplitude通過0的次數佔整個discrete signal間隔的比例。這個特徵已經被多運用在語音及音樂的資料分析，實作在本專題用來判斷波形擺盪頻繁程度。

iii. intergrated EEG : (1)

計算EEG在time domain的能量，每個epoch都不相同，藉由每個點amplitude的和代表這個epoch的能量程度。

- iv. Hjorth parameters(3)
分為activity、mobility、complexity。
activity represents the signal power, the variance of a time function.
mobility represents the mean frequency or the proportion of standard deviation of the power spectrum.
complexity represents the change in frequency.
- v. Detrended Fluctuation Analysis(1)
DFA(去趨勢波動分析)是一種判斷信號的統計自鄉性性質的方法。它可以用於分析類似長記憶過程的時間序列。
- vi. Shannon entropy(1)
The entropy gives a measure of signal disorder and can provide relevant information in the detection of some signal disturbs.

b. frequency analysis(24)

- i. non-parametric analysis(11)
[all power, delta power ratio, theta power ratio, alpha_low power ratio, alpha_high power ratio, beta power, ratio, gamma power ratio, DSI, TSI, ASI, spectral entropy]
we use welch's method to calculate PSD in every frequency. it is an approach to spectral density estimation. The method is based on the concept of using periodogram spectrum estimates, which are the result of converting a signal from the time domain to the frequency domain.
因為每個睡眠階段呈現的腦波頻率都不太一樣，因此分析腦波在每個頻率的能量是很重要的特徵。我們計算整個epoch的總能量，並且計算gamma, beta, low alpha, high alpha, theta, delta的power ratio，還有計算delta、theta、low alpha三者的比例比較分別是DSI、TSI、ASI，舉例DSI的計算方式是delta mean PSD/ (alpha low mean PSD + theta mean PSF)，希望藉由這三個參數可以明顯區分辨睡眠階段有很大特徵的alpha波、theta波、delta波。
spectral entropy 的計算方式是先 normalize the calculated PSD so that it can be viewed as a Probability Density Function.The Power Spectral entropy can be now calculated using a standard formula for an entropy calculation 。
- ii. parametric analysis
[peak_freq, q1_freq, cent_freq, q3_freq, 95_freq, sefIR, sefd, ssd, skew_P, kurt_P]
peak frequency就是指整個epoch中PSF最高的頻率，q1、cent、q3、95，分別代表用Spectral edge frequency去計算的頻率，算法是從最低頻率的能量開始累加達到總能量的25%、50%、75%及95%的頻率各為多少，再來計算sefIR是q1_freq - q3_freq，sefd是

high_freq - cent_freq, 然後ssd, skew, kurt分別是整個epoch power的standard、skewness和kurtosis。

iii. harmonic_parameter

[harm_fc, harm_fs, harm_p]

我們將每一個frequency及其PSD相乘, 除以整個epoch的能量, 當作central frequency, 再來計算bandwidth and the spectral value at center frequency。

c. time frequency(47)

i. DWT(41)

我們使用level=6的wavelet transform將訊號拆解為以下幾種頻段。[0-0.78][0.78-1.56Hz][1.56-3.125Hz][3.125-6.25Hz][6.25-12.5Hz][12.5-25Hz][25-50Hz], 並計算這些頻段的mean、std、averagepower、skewness、kurtosis, 最後計算兩兩頻段之間的平均值比例, 後一個頻段的mean值除以前一個頻段的mean值。

ii. EMD(6)

藉由Hilbert transform我們可以將訊號分為實部與虛部, 並且透過相位變化能夠計算瞬時頻率, 能量以Hilber spectrum表示, 然後計算不同頻段的總能量。頻段類型有[0.4Hz,1.55Hz],[1.55Hz,3.2Hz],[3.2Hz,8.6Hz],[8.6Hz,11.0Hz],[11.0Hz,15.6Hz],[15.6Hz,22.0Hz],[22.0Hz,30Hz], 再依據delta、alpha、beta及alpha/theta、delta/theta, 還有[0.4-1.55][11-15.6]兩種頻段能量和當作k-complexes+spindle, 最後每一個算出來的能量要取ratio除掉總能量。

d. complexity methods(13)

i. PFD(Petrosian Fractal Dimension)(1)

碎形維度, is a ratio providing a statistical index of complexity comparing how detail in a pattern (strictly speaking, a fractal pattern) changes with the scale at which it is measured.

ii. approximate_entropy(1)

an approximate entropy (ApEn) is a technique used to quantify the amount of regularity and the unpredictability of fluctuations over time-series data

iii. hurst_exponent(1)

the Hurst exponent is used as a measure of long-term memory of time series. It relates to the autocorrelations of the time series, and the rate at which these decrease as the lag between pairs of values increases

iv. Maximum minimum distance(5)

先把訊號透過band pass filter分成delta、theta、low alpha、high alpha、beta五種波段, 再分別將這些波段每個epoch分成30個

segment, 每個segment計算最大值與最小值的距離, 其中包含 amplitude及時間的訊息, 然後30個數值加起來總和。

v. EnergySis(5)

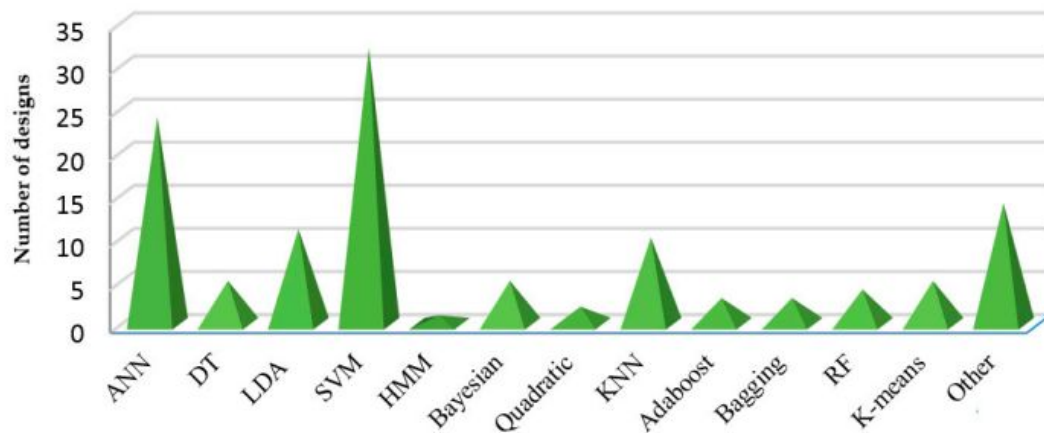
把分成五個頻段的波的passband中點當作這個波的頻率, 然後假設波長是100, 計算出每個頻段的波速率, 再將每個amplitude電壓平方成上速率當作能量分析。

2. feature selection

feature selection的方法有很多種, 以前paper使用的方式也都不相同, 本專題並未事先對feature做selection, 而是透過之後的model來判斷feature importance。

3. classification method

本專題使用XGboost當作訓練model, 由於過去沒有人使用此方法當作method, 如下圖統計以前使用的方法, 大多數都是使用SVM。



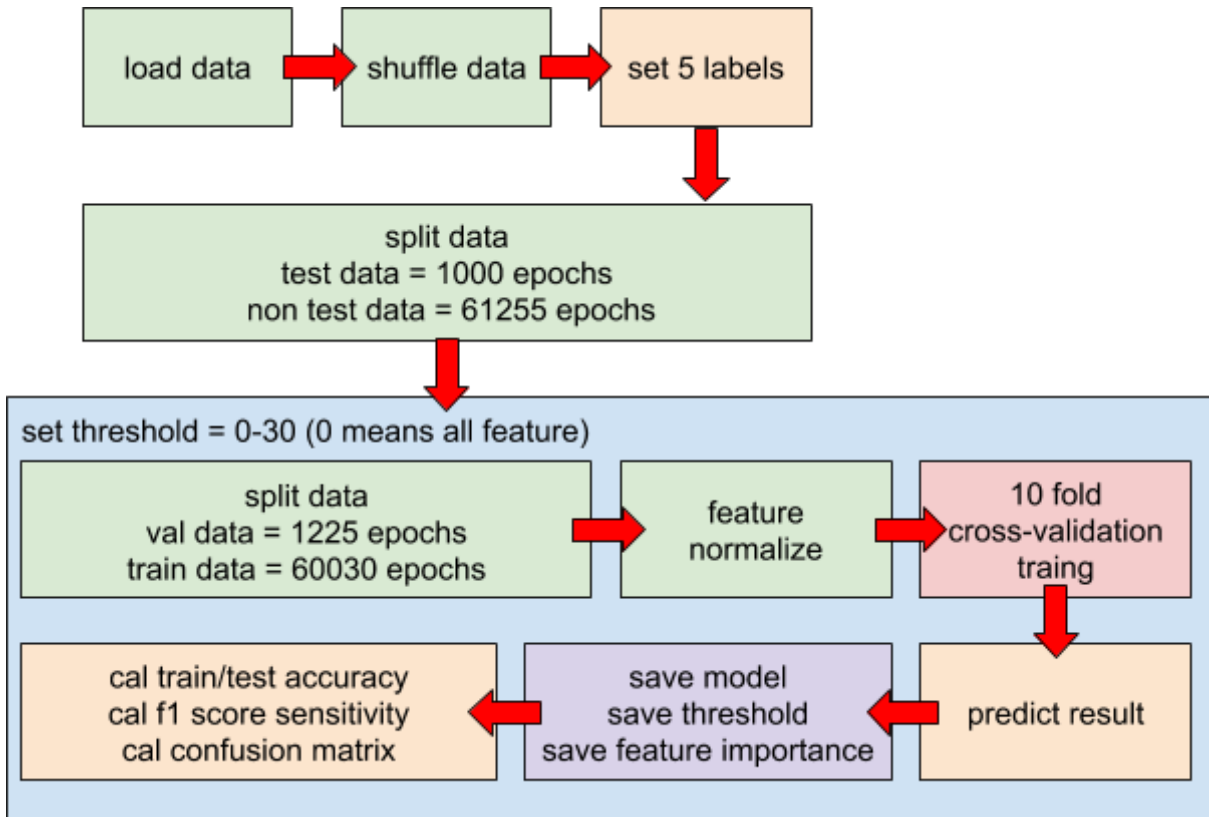
因此我們想嘗試看看效果為何, 並且透過XGboost幫我們計算feature importance, 再以importance較高可是較少的feature進行training, 來比較model未經selection的結果。以下是對xgboost的簡單介紹:

- method introduction

XGBoost的全稱為eXtreme Gradient Boosting, 是GBDT的一種高效實現。GBDT(Gradient Boosting Decision Tree) 又叫 MART (Multiple Additive Regression Tree), 是一種疊代的decision tree算法, 該算法由多棵decision tree組成, 所有tree的結論累加起來做最終答案。它在被提出之初就和SVM一起被認為是generalization較強的算法。GBDT的核心在於, 每一棵tree學的是之前所有tree結論和的殘差, 這個殘差就是一個加預測值後能得真實值的累加量。與random forest不同, random forest採用多數投票輸出結果; 而GBDT則是將所有結果累加起來, 或者加權累加起來。相對GBDT, XGBoost有做一些進步, 比如說為了避免overfitting, 有多使用regulization; loss function使用二階泰勒展開式, 精度更高; 每棵樹的節點優化, 分裂時原本是通過計算分裂後的某種值減去分裂前的某種值, 從而得到增益, 但多增加參數限制tree的生成, 以及leaf更為平滑。

整體上，經過優化及其易用性不太需要compile，是目前GBDT體系中最受歡迎的工具。

- training flow chart



- parameters

因為XGBClassifier及做cross-validation的參數較多，故詳細列出。

```
xgb = XGBClassifier(  
    learning_rate=0.1, n_estimators=1000,  
    max_depth=5, min_child_weight=1,  
    gamma=0, subsample=0.8,  
    colsample_bytree=0.8, objective='multi:softmax',  
    nthread=4, scale_pos_weight=1, n_jobs=24)
```

```
xgb.cv(  
    xgb_param,  
    dtrain,  
    num_boost_round=alg.get_params()['n_estimators'],  
    nfold=cv_folds,  
    metrics='mlogloss',  
    early_stopping_rounds=early_stopping_rounds,  
    verbose_eval=True)
```

- threshold
由於XGBoost會幫我們把每個feature打分數，因此可以設threshold如下圖。我們從一個feature慢慢再加入次高的feature一起做training，並期望可以看到少量的feature就可以達到高的accuracy，並不需要全部99個feature都當作training的參數。

```
selection = SelectFromModel(xgb1new, threshold=thresh, prefit=True)
select_X_train = selection.transform(X_train)
select_X_test = selection.transform(X_test)
feature_size = select_X_train.shape[1]
```

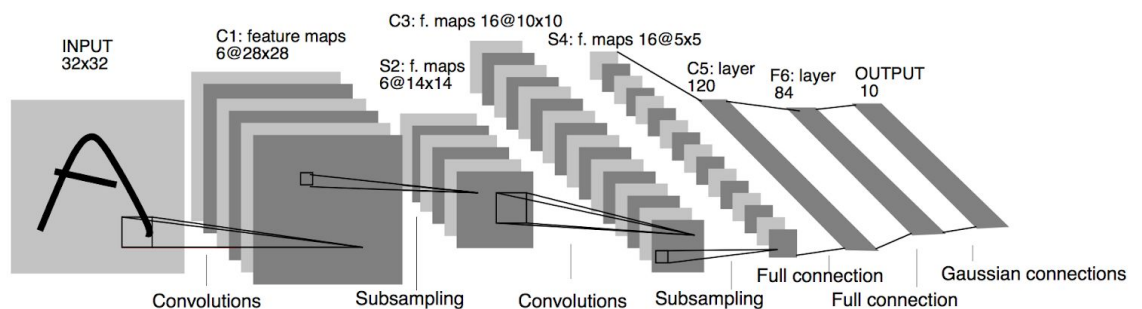
- plot importance
可以藉由XGBoost內置的API函式，將每個feature的分數以長條圖的方式畫出來，藉此判斷本專題的實驗結果哪些feature具有相當好的訊息。我們圖是只畫出前30個分數高的feature，詳見之後的result部分。

七、Deep Learning

1. Introduction

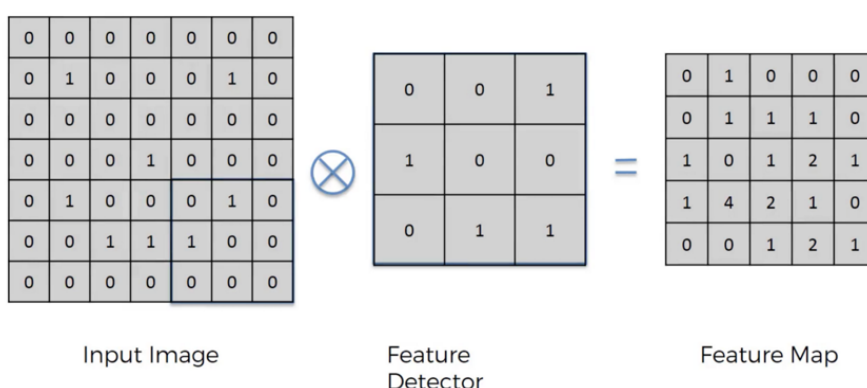
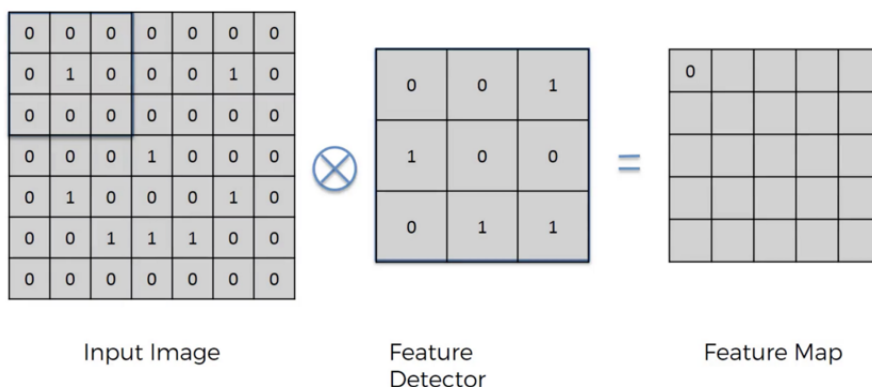
Convolutional Neural Network (CNN)是Deep Neural Network最常見的架構之一，在image recognition、pattern recognition等領域被廣泛地應用。傳統的Fully-connected network最大的問題在於，由於它的input必須是一維的，因此需要先將圖片攤平(flatten)才能輸入network之中，通常一張影像資料包含了水平、垂直、color channel等三維資訊，攤平拉成一維後會導致資料的部分特徵、形狀資訊無法被保留起來，而CNN可以直接將三維影像當作input，保留圖像的空間排列順序，使得這些重要特徵得以保存。

CNN較傳統的DNN多了Convolutional及Pooling (subsampling)兩層layer，用以維持形狀資訊並減少運算時間及參數量，下圖是經典的LeNet的架構圖，由Yann LeCun於1998年提出，被公認為CNN的始祖，以下分別介紹CNN架構裡的三種layer。



- Convolutional layer
Convolution的原理是透過一個事先人為訂定尺寸(kernel size)的filter，由左而右、由上而下依序滑動和圖片的pixel做convolution，來取得圖像中各局部特徵作為下一層的輸入，而不同大小的size能讓network學到圖形當中不同的特徵，下圖用簡單的例子來做說明，input image的大小為

7*7, 我們使用一個 kernel size 為 3*3 的 filter 來做 convolution, 結果如下 :

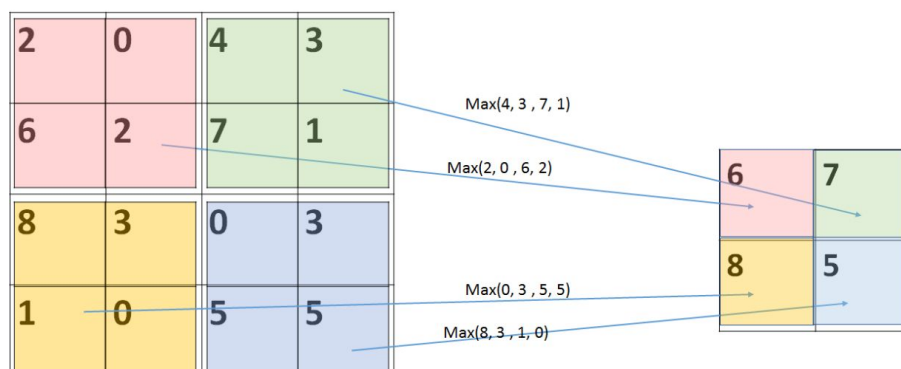


做完convolution後, 我們會再加上一個 activation function, 通常為 ReLU(Rectified Linear Unit), Leaky ReLU, Tanh, Sigmoid 等 non-linear function, 目的是讓 model 能包含更多非線性元素、並將輸出固定在某一區間(通常為(0,1)或(-1,1)之間), 使得在backpropagation更新參數時能更加穩定、得到更佳的分類結果。

- Pooling layer

在做convolution時, filter會對整張圖片滑動, sliding window會有過多重疊的範圍, 出來的值可能會有些冗餘。Pooling的主要功能為使輸入的圖片尺寸縮小一半, 以減少feature map的維度並過濾這些convolution產生的局部冗餘資訊, 保留最重要的特徵, 同時, 透過pooling可以減少後續幾層layer所需的參數量, 加快model運算的效率, 並減少over-fitting發生的可能性。此外, Pooling layer具有良好的抗雜訊功能, 舉例而言, 若圖像中某些pixel 在鄰近區域有微小偏移或差異時, 對Pooling layer的輸出影響不大, 結果仍是不變的。

常見的Pooling方式主要有兩種: Max-Pooling、Average-Pooling, 下圖以 Max-Pooling來做說明:

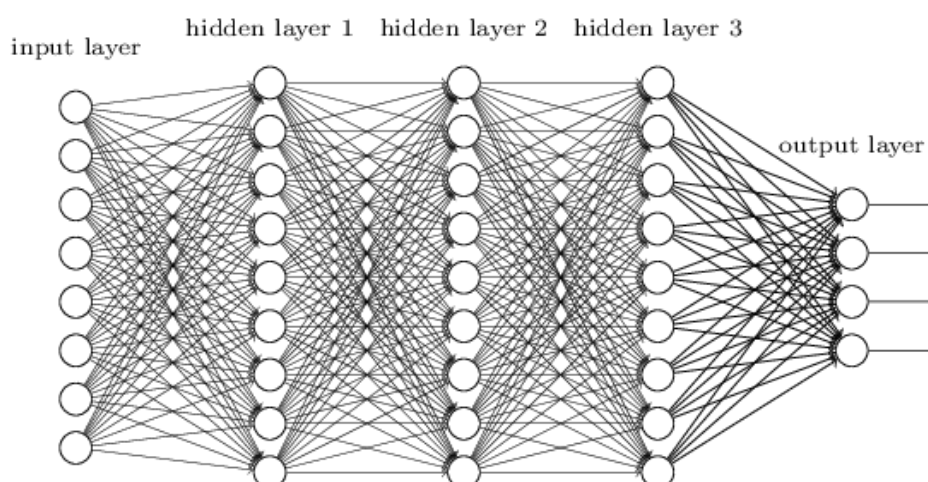


最常用的Pooling kernel size是2×2，Stride為2，這個設定可縮小一半尺寸並減少75%運算量，大大增加CNN的運算效率。

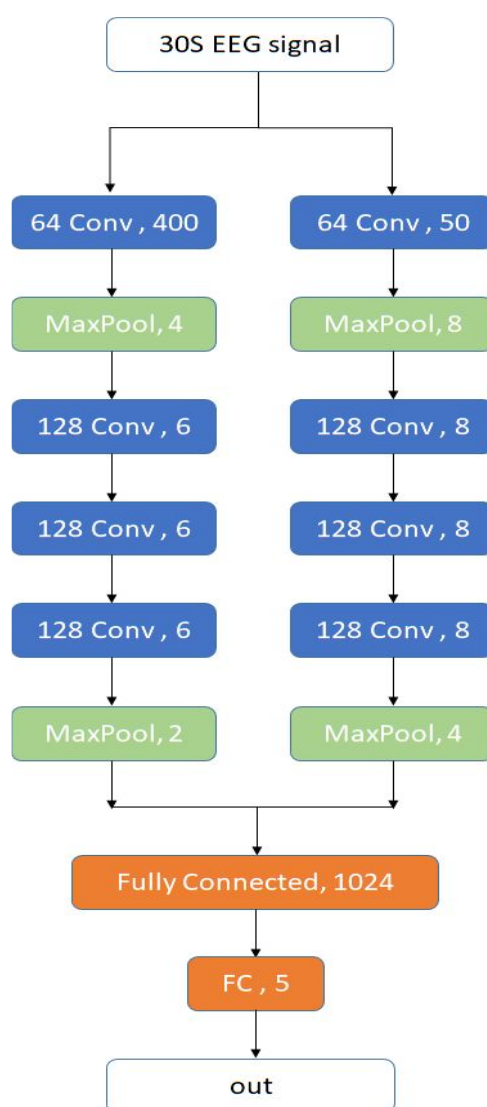
- Fully-Connected layer (FC)

Fully-Connected layer通常會接在Convolution或Pooling層後面，“分類”的動作也是在這個階段進行。如同之前所提，FC的input必須要是一維的，透過將每個神經元與上層神經元之間彼此互相連接，各個連結都有其獨立且相異的權重(weight)和偏差(bias)，如下圖所示，單一神經元neuron的transform可以表示為 $y = w * x + b$ ，其中w和b分別為此neuron的weight以及bias，訓練的目標就是找出最適當的w和b，在此和先前提到的一樣，為了避免訓練的結果出來是線性，我們會在公式外再加上一個non-linear的activation function，而在output layer前，一般情形下會使用一個softmax function，將輸出限制在(0,1)之間，這個數值可以視作這個input屬於某一個class的機率，機率最大的即為model判斷的類別。

另一方面，Full connected layer耗用相當多的運算資源，算是這個架構最大的缺點，model中大部分的參數量都在此層，很吃電腦的記憶體。舉例來說，現今學術上常用來抽取圖像feature的VGG-16，總共參數量為138,357,544，光是model裡三層FC layer的參數就高達123,642,856，顯示出FC層在model佔有舉足輕重的角色。



2. Model architecture



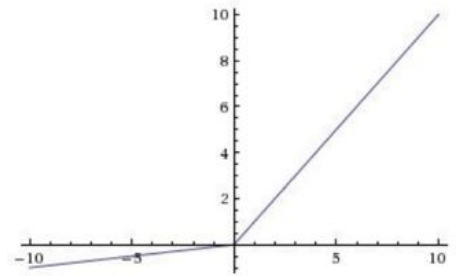
上圖為我們本次專題裡所建構的model network架構。首先直接將EEG raw data當作input，分別經過兩個size不同的CNN block後concatenate起來接到後面的FC來做classification，最後output為屬於這五個class的機率。

圖中64 Conv, 50代表的是該層convolutional layer共有64個filter，每一個filter size為 1×50 。之所以要用兩個CNN block原因在於，我們藉由設計這兩個block參數，來讓兩個CNN學到data中不一樣的資訊：左邊CNN第一層的size較大(400)，做convolution時會更有大局觀，看到更大範圍的data，這可以使它學到大範圍有關頻率的資訊，而右邊CNN第一層size較小(50)，則是讓它學到局部較temporal的資訊，也就是某些特定的EEG pattern，我們希望藉由這樣個設計能讓model學到更多元的feature，後面result的部分我們會將這兩個CNN自己學到的資訊來做視覺化的呈現。

3. Training settings

以下將我們training時所訂定的一些model參數及設定來做簡單說明：

1. Optimizer : Adam, with learning rate=0.0001
2. Loss function : Categorical CrossEntropy loss
3. Activation function : Leaky ReLU (如右圖)
4. Total 42308 data with corresponding labels, we split it into three pieces :
38000 for training set, 2308 for validation set,
2000 for testing set.
5. batch_size=128
6. training epochs = 150



八、Results

1. traditional learning

- accuracy

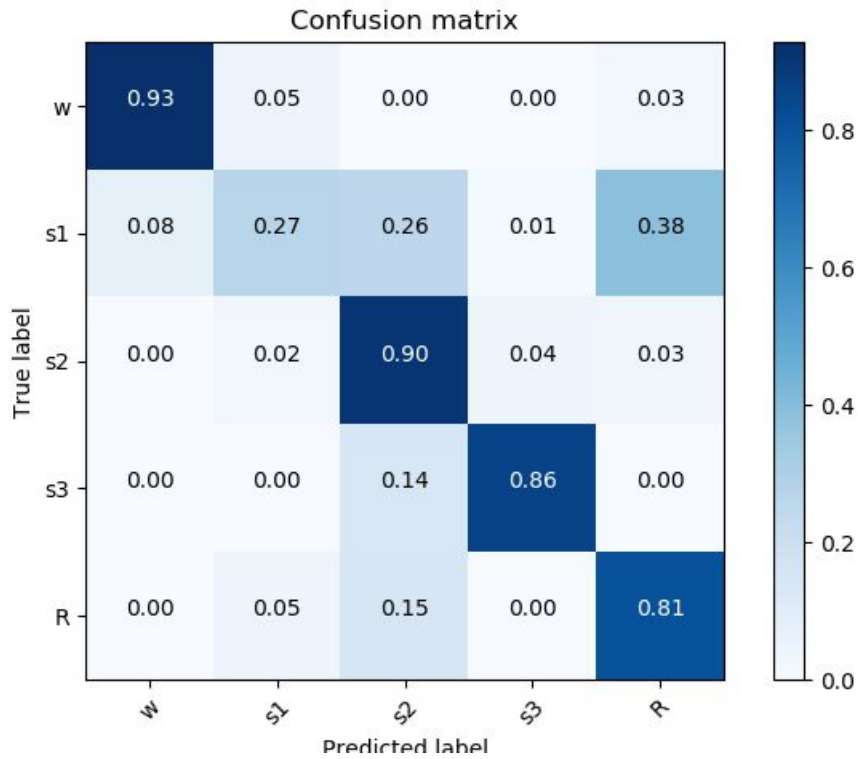
從表格中可以發現我們本專題對99個feature使用XGBoost的準確率結果，accuracy約為83.5%左右，其中f1 score及sensitivity為76%及75.3%，另外training accuracy有到0.953，可了解我們training可以成功有結果，但是會有一點overfitting的現象

feature size = 99		
train accuracy	train f1 score	X
0.953	0.940	X
val accuracy	val f1 score	val sensitivity
0.822	0.750	0.743
test accuracy	test f1 score	test sensitivity
0.835	0.760	0.753

- confusion matrix

縱軸是真實的label，橫軸是model預測的label，可以清楚看到data較多的wake及stage2都有很好的效果約90%以上的準確率，另外stage3及REM的準確率分別為86%及81%，因為這兩個階段的特徵沒有wake來的明顯。再來發現stage1的準確率只有27%，符合和過去文獻比較的結果，要判斷stage1較難，必須要能和REM做區別，兩個階段的feature相似，而且又因為和部分無明顯特徵的stage2很像，造成判斷stage1是專題的bottleneck

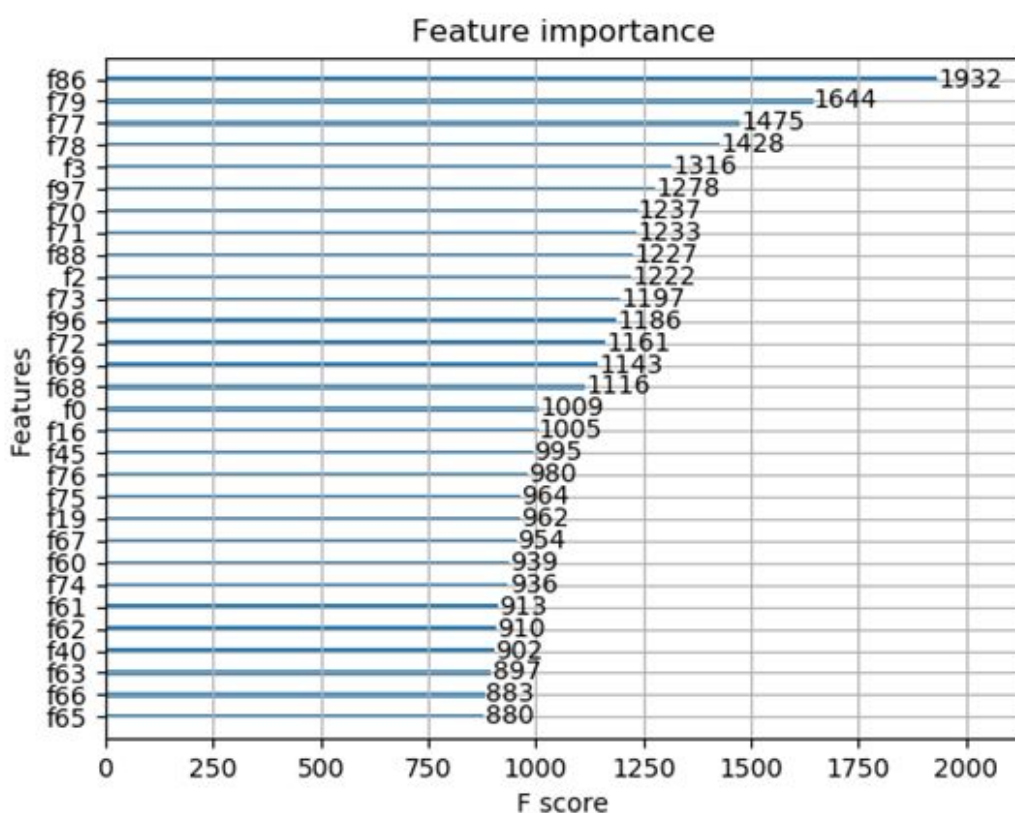
。



- feature importance

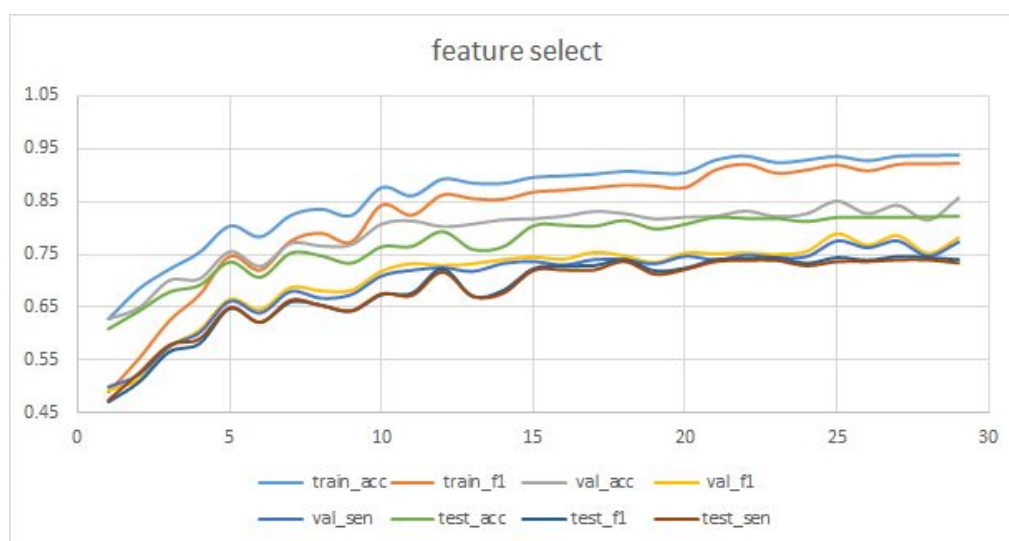
XGBoost在training的過程會對每個feature算分數(如下圖), 因為只有部分的feature會對regression tree有幫助, 因此我將前15個的結果列出來(如下表格), 發現多與小波轉換wavelet transform有關, 且有許多都是focus在cA6這個很小的頻段上, 推測是因為這與分析深層睡眠的波段有關, 然後高頻段的alpha波分析則是利用esisenergy的算法很有幫助。

PFD (86)	Esisen high alpha (97)	DWT kurtosis cA6(73)
DWT mean cA6/cD5 (79)	DWT std cA6 (70)	Esisen low alpha (96)
DWT mean cD4/cD3 (77)	DWT avgPower cA6 (71)	DWT skewness cA6(72)
Dwt mean cD5/cD4 (78)	Hurst exponent(88)	DWT mean cA6(69)
Kurtosis (3)	Skewness (2)	DWT kurtosis cA6(68)



- feature select

從feature importance可以看出每個feature有不同分數，因此依照分數可以有不同的threshold，再select不同數量的feature做training看效果如何。下圖是train、validation和test的圖，可以發現train的值都較高，因為我們的model已經fit training data；然後是val和test兩者的accuracy；再來為兩者的f1 score，f1 score會較低是因為他有考慮label的balance，以免model是用亂猜得出結論；最後是sensitivity。

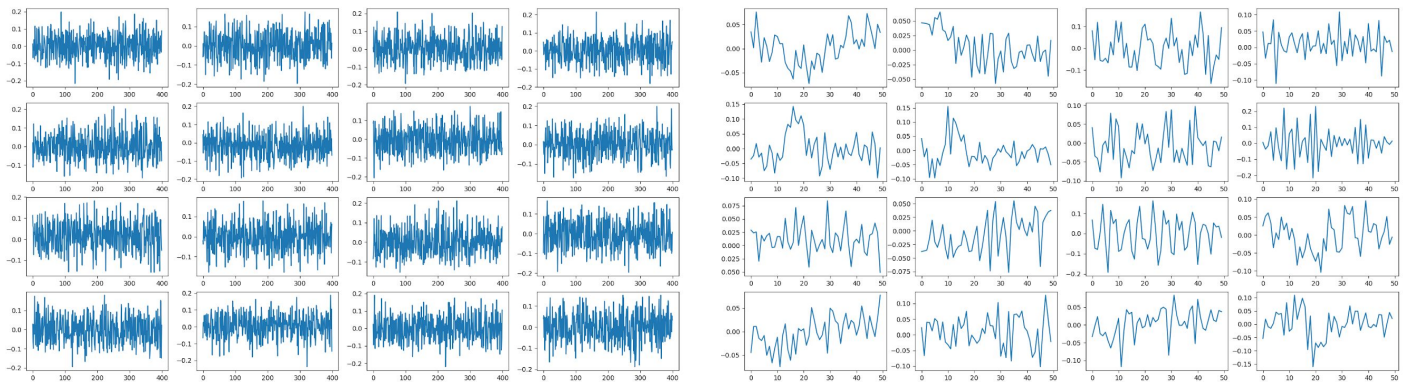


觀察feature size, 可以看到剛開始因為feature數量較少accuracy較低, 但慢慢增加數量會使得準確率有所提升, 而且不一定要99種feature都做training效果會最好, 只要具有一定數量的重要特徵就可以達到不錯的效果。下表格是其中最高的準確率, 當我們選取前25個重要的feature準確率大概就有82%左右, 代表不需要所有的feature, 只需要重要的feature

feat size	train acc	train f1	val acc	val f1	val sen	test acc	test f1	test sen
25	0.936	0.918	0.850	0.789	0.776	0.82	0.744	0.737
99	0.953	0.940	0.822	0.750	0.742	0.835	0.76	0.753

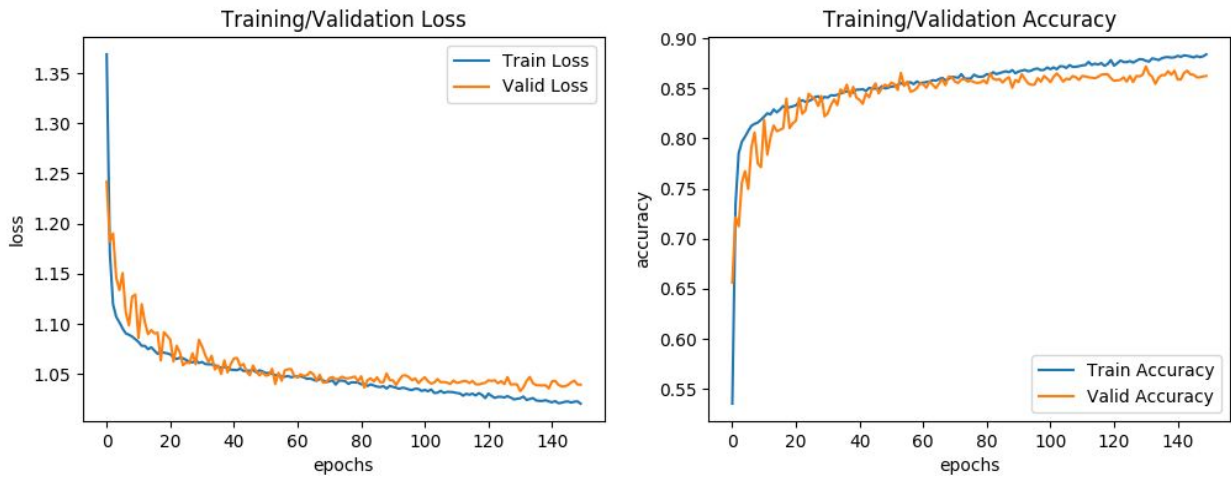
2. Deep learning

- feature map



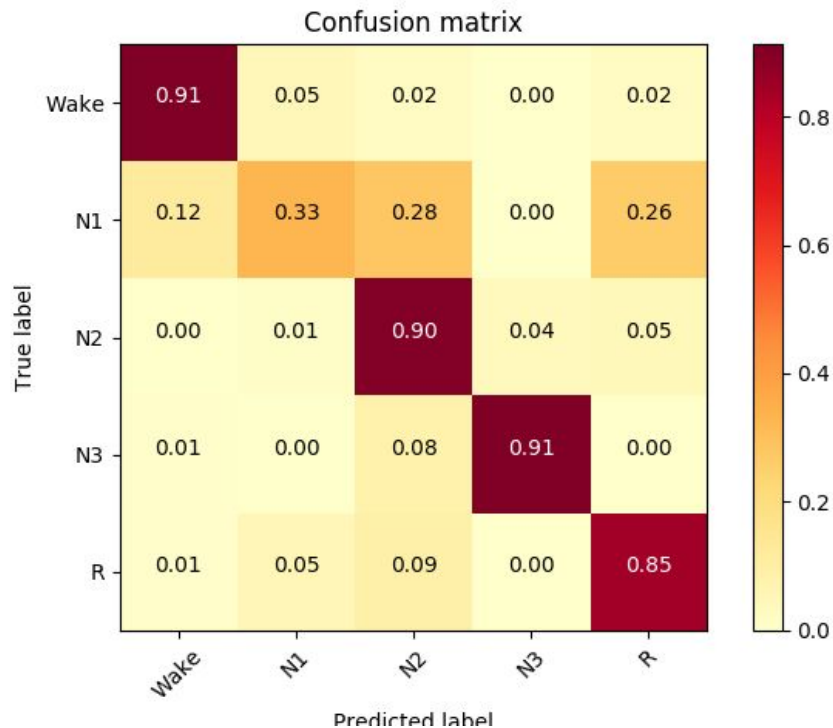
左圖為小size的CNN第一層前16個filer所繪出的feature map, 右圖則為大size CNN 繪出的feature map, 可以看出右圖的feature比較sparse, 顯示出model學到的是較大範圍頻率的資訊; 而左圖的feature看似較為雜亂, 判斷應該是當下時間點的波形pattern, 值得一提的是, 我們並不會知道CNN學到的feature到底是什麼、具有什麼樣的物理意義, 這也是CNN神奇的地方之一, 它有辦法自己學習、抽取feature來做複雜的分類問題。

- learning curve



我們總共train了150個epoch， loss & accuracy 分別為左右兩圖， 我們可以從圖中觀察出， 在 training 初期 training 和 validation loss 穩定快速下降， accuracy也急速上升， 大約在40~50 epoch左右即開始收斂， training loss 仍持續下降， 而validation則不斷震盪， 80 epoch之後開始有一點 overfitting的現象發生， 最後validation accuracy最高為0.8718， testing accuracy為0.86 (1720/2000)。

- confusion matrix



縱軸是真實的label，橫軸是model predict 的 label，從結果來看，Wake, N2, N3, REM都能得到極高的準確率，stage 1~3, REM的準確率也比前面所使用的xgboost略高，而N1準確率同樣仍然不甚理想，只有33%，推測是因為stage 1 並沒有明顯的波形feature，導致model不容易將此和stage2、REM做出區隔，這部分是未來需要克服的問題。

九、Discussion

1. Data imbalance影響performance

由於睡眠的每個階段長短不同，會造成imbalance的問題，尤其stage 1的以SC* file和ST* file加起來數量只有其他階段的一半左右，很難使model可以判斷精準。除此之外，stage 1與REM都是頻率較大，振幅較小的特徵，兩者主要差別是在眼動的現象，但本專題只有針對EEG訊號當作data，並未使用EOG眼電訊號。

解決方法我們有想到兩種，一種是oversampled，就是把data較少的類別複製更多份epoch，使得每一個label的data數量一致，期望可以解決因為不平均而有亂猜f1 score較低的狀況。另一個方法是使用XGBoost的時候可以多考慮EOG的feature，用來分辨stage 1及REM的差別。

2. Normalize效果

因為每個受試者個體本身就存在差異，我們嘗試看看是否將每個受試者都對自己的生理訊號做normalize，以消除個體差異。但最後效果不盡理想，推測是因為原本的波會整個變形，變成不是從人體量出來的真實生理訊號，這樣子取feature反而會失真。因此最後我們是對feature做normalize，而非個別每個file的EEG訊號，但在使用CNN或是XGBoost效果也沒有說差很多，推測是因model自己就會學到對feature有標準化的作用。

3. 改進空間(LSTM)

除了CNN之外，deep learning 還有另外一種常用且效果良好的model，也就是Recurrent Neural Network (RNN)，RNN具有記憶的特性，不同於CNN或DNN只能取得當下時間點的data，RNN能將過去所看到的時間序列data重要的部分一併記錄下來，我們所使用的EEG input即符合這樣的條件，當下時間點也可能和前後時間有所關聯，因此若使用RNN應能得到不錯的結果，常用的RNN module有LSTM和GRU兩種，未來我們能改進的方式為將CNN和RNN結合，透過CNN來取EEG波形的feature，再利用RNN將前後時間點的關係combine起來，來做最後的classification。

十、Application - alarm clock

1. mindwave mobile

首先我們嘗試用此產品附的app來進行實作，但後來發現我們沒辦法從中取出data，這樣變無法和我們的model進行結合，因此只好徒手把data從腦波儀中挖出來。一開始碰到的問題是windows可以用藍芽配對，並且可以設置要input和output的port是什麼，但是沒有辦法跑程式的部分，而MAC的設計又很不user-friendly，花了很多時間之後終於找到了mac在根目錄的地方有設置藍芽連接的port，再配對之後會發現。可以看到下圖左邊為一般的藍芽的port，右邊則是已經配對完成的樣子。

```
ip77-120:~ Ken$ ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port /dev/cu.MindWaveMobile-SerialPo
ip77-120:~ Ken$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port /dev/tty.MindWaveMobile-SerialPo
```

接著我們用python import藍芽的套件去做連結，因為這個腦波儀藍芽連線不是很穩，一開始我們用app連接之後，確定連線穩定，再執行指令，卻怎麼樣都連不上，正當想放棄的時候，app的連線中斷了，而我們這邊卻連上了，才發現原來和腦波儀的port只有一個，如果用app連接之後，就不能再用別的方式連結了，雖然聽起來很合理，但一開始怎麼也沒想到，也算是學到了一課。

```
import numpy as np
headset = mindwave.Headset('/dev/tty.MindWaveMobile-SerialPo')
#headset = mindwave.Headset('/dev/tty.MindWave')
cflag = 0
```

再來，我們照著產品提供的communication protocol中說明的封包傳遞方式，寫出了解碼的程式。

```
import select
import serial
import threading

# Byte codes
CONNECT          = '\xc0'
DISCONNECT       = '\xc1'
AUTOCONNECT     = '\xc2'
SYNC             = '\xaa'
EXCODE           = '\x55'
POOR_SIGNAL     = '\x02'
ATTENTION        = '\x04'
MEDITATION      = '\x05'
BLINK            = '\x16'
HEADSET_CONNECTED = '\xd0'
HEADSET_NOT_FOUND = '\xd1'
HEADSET_DISCONNECTED = '\xd2'
REQUEST_DENIED  = '\xd3'
STANDBY_SCAN    = '\xd4'
RAW_VALUE        = '\x80'

[ 8]: 0x00 // (2/3)
[ 9]: 0x94 // (3/3) End Delta bytes
[10]: 0x00 // (1/3) Begin Theta bytes
[11]: 0x00 // (2/3)
[12]: 0x42 // (3/3) End Theta bytes
[13]: 0x00 // (1/3) Begin Low-alpha bytes
[14]: 0x00 // (2/3)
[15]: 0x0B // (3/3) End Low-alpha bytes
[16]: 0x00 // (1/3) Begin High-alpha bytes
[17]: 0x00 // (2/3)
[18]: 0x64 // (3/3) End High-alpha bytes
[19]: 0x00 // (1/3) Begin Low-beta bytes
[20]: 0x00 // (2/3)
[21]: 0x4D // (3/3) End Low-beta bytes
[22]: 0x00 // (1/3) Begin High-beta bytes
[23]: 0x00 // (2/3)
[24]: 0x3D // (3/3) End High-beta bytes
[25]: 0x00 // (1/3) Begin Low-gamma bytes
[26]: 0x00 // (2/3)
[27]: 0x07 // (3/3) End Low-gamma bytes
[28]: 0x00 // (1/3) Begin Mid-gamma bytes
[29]: 0x00 // (2/3)
[30]: 0x05 // (3/3) End Mid-gamma bytes
[31]: 0x04 // [ATTENTION] eSense
[32]: 0x0D // eSense Attention level of 13
[33]: 0x05 // [MEDITATION] eSense
[34]: 0x3D // eSense Meditation level of 61
```

由於封包傳遞是用bytes的方式，所以要把這些16進位的數再經過處理，而現在python的函式有點不太支援這種decode方式，因此也是費了一些功夫才成功。再來碰到了一些問題，這些protocol還有網站上都沒有看到這些數據的詳細描述，連單位都沒有，只有說是怎麼解碼，但我們解出來和我們的database的data不

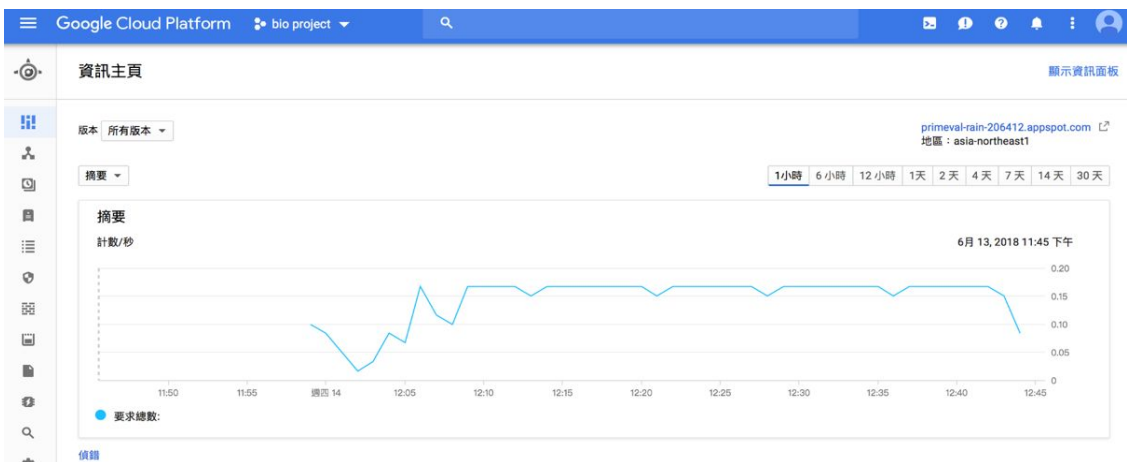
太吻合，因此我們做了一些調整讓他的值跟我們的值比較相近，而後來去做 predict時結果也還算吻合，因此就先繼續使用。

下圖為我們程式跑的結果，最下面會print出raw value，我們設定每0.01秒access一次腦波，這樣才可以吻合我們的database的30秒3000筆數據，但由於連線會有不穩定的狀況，可以看到上面常常有connection break，就是中間訊號不太好的狀況，這些值我們就不取，所以會取到約40秒才會有3000筆數據

```
16:34 Jing-Chengs-MacBook-Pro@jingcheng:~/Desktop/biopproject/ASSC$ python3 test.py
Initializing....
running.....
Connecting...
Connected.
status=connectedumber 130
connection break...
status=connectedumber 3860
connection break...
status=connectednumber 642
connection break...
status=connectednumber 810
connection break...
status=connectedumber 9770
connection break...
status=connectednumber 1143
connection break...
status=connectedumber 12216
connection break...
status=connectednumber 1300
connection break...
status=connectedumber 146657
connection break...
status=connectedumber 15453
connection break...
status=connectedumber 16233
connection break...
raw_value: -97 number 19098
```


2. android app

我們predict完一個值之後，要怎麼呈現在app上面？我們研究後發現APP Inventor有和Google Cloud Platform合作，有一個平台可以讓我們從本機端寫值進去，再讓app去那個網站上面把值取下來，但有個問題是這個平台是一個開放式平台，會有很多其他的資訊或是我們的資料會有遺失的風險，因此我們要自己建一個平台，使用的就是Google Cloud Platform裡面的App engine。我們首先要在平台上建立專案，取得專案ID。



然後下載App Engine for Python以及GoogleAppEngineLauncher還有app inventor 的 customtinywebdb, 跑一些程序和安裝一些套件之後, 把這個專案deploy到雲端, 接著我們就可以有一個很簡單的key-value儲存的網頁, 很方便取值還有還儲存資料, 作為一些實作即時成果展現的題目時應該都派得上用場。

App Inventor (TinyWebDB) Web Dat



This web service stores and retri this service using the TinyWebDI

Search database for a tag

Tag:

Get value

Returned as value to TinyWebDB component:

Store a tag-value pair in the database

Tag:

Value:

Store a value

Key	Value	Created (GMT)	
stage	tensor(0)	June 13, 2018, 3:06 p.m.	Delete

接著就是app設計的部分, 由於是要實作一個可以設定最早和最晚起床時間的鬧鐘 因此在時間設定上不像以往鬧鐘那麼簡單, 用一個24小時制的時間, 等於設定的就可以播放音樂, 在這裡我們要能夠去檢測時間的先後順序, 而我使用的方法是把時間換成24小時制, 然後全部換成分鐘, 然後如果最晚起床時間的值比最早起床的小的話, 也就是可能一個晚上11點一個凌晨1點這種狀況, 那就會再加24小時的時間上去讓他是一致的。

```

if (compare texts (get global PAM) = " PM ")
then
  if (get global 幾點1) < 12
  then
    set global timenow1 to (get global 幾點1) + 12
  if (get global 幾點2) < 12
  then
    set global timenow2 to (get global 幾點2) + 12
else
  if (get global 幾點1) <= 12
  then
    set global timenow1 to 0
  else
    set global timenow1 to (get global 幾點1)
  if (get global 幾點2) <= 12
  then
    set global timenow2 to 0
  else
    set global timenow2 to (get global 幾點2)
if (get global 幾點1) > (get global 幾點2)
then
  set global timenow2 to (get global 幾點2) + 12
if (get global 幾點1) > (get global hour) and (get global hour) = 0
then
  set global timenow to ((get global hour) + 24) * 60 + (get global min)

```

下圖則是結合App engine的部分，給他我們deploy好的網址，設定clock rate，他就會定期地去取值出來

```

when Clock1.Timer
do
  set TinyWebDB1.ServiceURL to "http://primeval-rain-206412.appspot.com"
  call TinyWebDB1.GetValue tag "stage"
  set Player1.Source to "music.wav"
  set global hour to call Clock1.Hour instant call Clock1.Now
  set global min to call Clock1.Minute instant call Clock1.Now

when TinyWebDB1.WebServiceError
message
do
  set stage_num.Text to "error"

when TinyWebDB1.GetValue
tagFromWebDB valueFromWebDB
do
  set global stage_num to get valueFromWebDB
  set stage_num.Text to get valueFromWebDB

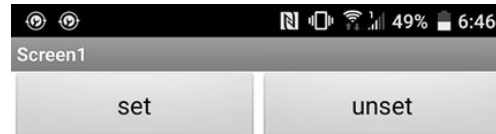
then
  set global timenow to (get global hour + 24 * 60 + get global min)
else
  set global timenow to (get global hour * 60 + get global min)

set global time1 to (get global timenow * 60 + get global 幾分1)
set global time2 to (get global timenow * 60 + get global 幾分2)

if
  (get global timenow >= get global time1) and (get global time2 >= get global timenow)
then
  if
    get global model = true
  then
    if
      (get global stage_num = "W" or get global stage_num = "R")
    then
      call Player1.Start
  
```

上圖則是比較，看現在的時間有沒有在指定的範圍內，如果有而且現在的stage也是在wake或是快速眼動期，也就是剛過完一個週期的話，我們便會響，那如果時間是等於最晚起床的時間，也會直接叫他起床。

app的版面設計就蠻陽春的，就是時間設定好之後，要按set才会有設定，unset則相反基本上操作和一般鬧鐘無異，當開始響時最下方會跳出stop的按鈕按完又會消失不見



stage:



十一、conclusion

藉由EDF[expanded]database，我們使用兩種machine learning方式來實作automatic sleep stage classification，第一個是實作traditional learning，先feature extraction出99種features，然後使用XGBoost model實作training；第二個是實作deep learning，直接將raw data當作input給CNN model實作training。兩種方式的準確率分別為0.835與0.86，但是我們兩種model還是無法解決stage1的辨識，容易將其判斷成REM以及stage2。最後我們將model應用在手機app鬧鐘上面，目標為使用者以睡眠階段當作依據，安排最適合的起床時間，解決起床會精神差心情不好的問題。

十二、reference

- [1]Şen, Baha, et al. "A comparative study on classification of sleep stage based on EEG signals using feature selection and classification algorithms." *Journal of medical systems* 38.3 (2014): 18.
- [2]Dong, Hao, et al. "Mixed neural network approach for temporal sleep stage classification." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.2 (2018): 324-333.\
- [3]Fehrmann, Elizabeth Ann. *Automated sleep classification using the new sleep stage standards*. Rochester Institute of Technology, 2013.
- [4]Li, Yi, et al. "Sleep stage classification based on EEG Hilbert-Huang transform." *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. IEEE, 2009.
- [5]Sanders, Teresa H., Mark McCurry, and Mark A. Clements. "Sleep stage classification with cross frequency coupling." *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 2014.
- [6]Aboalayon, Khald Ali I., et al. "Sleep stage classification using EEG signal analysis: a comprehensive survey and new investigation." *Entropy* 18.9 (2016): 272.
- [7]N. Liu, Z. Lu, B. Xu and Q. Liao, "Learning a convolutional neural network for sleep stage classification," *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Shanghai, 2017, pp. 1-6.
- [8]Supratak, Akara, et al. "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.11 (2017): 1998-2008.
- [9] Nakamura, Takashi, et al. "Complexity science for sleep stage classification from EEG." *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.
- [10] Alickovic, Emina, and Abdulhamit Subasi. "Ensemble SVM method for automatic sleep stage classification." *IEEE Transactions on Instrumentation and Measurement* 67.6 (2018): 1258-1265.
- [11]Şen, Baha, et al. "A comparative study on classification of sleep stage based on EEG signals using feature selection and classification algorithms." *Journal of medical systems* 38.3 (2014): 18.
- [12]Khalighi, Sirvan, et al. "Automatic sleep staging: a computer assisted approach for optimal combination of features and polysomnographic channels." *Expert Systems with Applications* 40.17 (2013): 7046-7059.
- [13]Lajnef, Tarek, et al. "Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines." *Journal of neuroscience methods* 250 (2015): 94-105.
- [14]Chambon, Stanislas, et al. "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (2018).