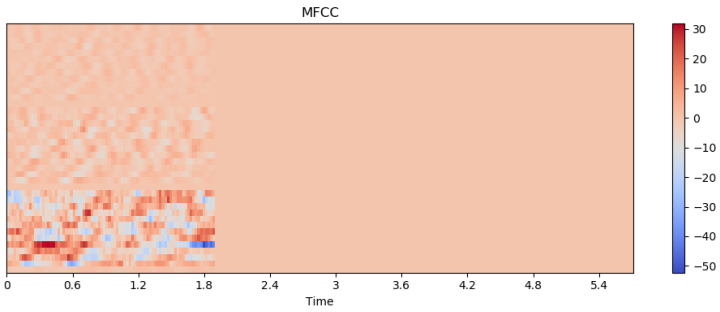
ML final Report

- 選擇題目：Listen and Translate
- 隊名：NTU_r06922113_危險
- 隊員名單：R06946013 謝宗翰 R06922113 陳宣伯
- 成員分工：
  - 資料前處理　　：陳宣伯、謝宗翰
  - 討論模型架構，決定實驗方法　：陳宣伯、謝宗翰
  - 實作 Retrival model　　：陳宣伯
  - 實作 CNN model　　：謝宗翰
  - github code 整理上傳　　：陳宣伯
  - 撰寫 Report　　：陳宣伯、謝宗翰
- 資料前處理：
  - Mfcc



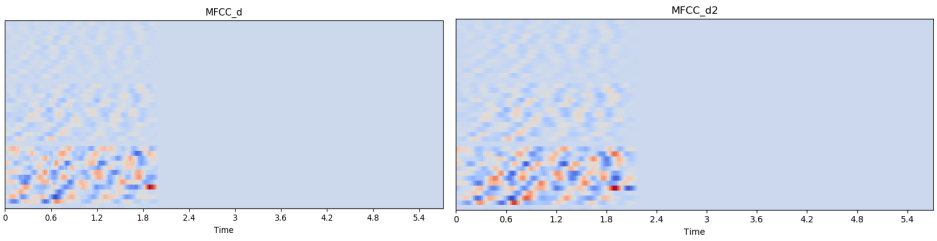
model 的 input 為每個台語句子的音訊檔轉換為 mfcc 的參數，為一個時間長度 x 39(mfcc 維度)的矩陣。

在 train data 中時間長度最長的資料為 246（約 5.72 秒），所以我們將每筆資料 reshape 成(246x39)，不足的部分補 0 處理。

有嘗試過使用 librosa 所提供的 mfcc-delta 對 mfcc 資料進行處理，詳見：

https://librosa.github.io/librosa/generated/librosa.feature.delta.html?highlight=delta

產生 mfcc-delta 及 mfcc-delta2，視覺化如下圖：



試圖藉此讓 model 觀察到更多有關音訊的訊息，但因為如此會訓練

資料大小變為原本的 3 倍，model 訓練時間會拉得太長且進步幅度不明顯，在我們硬體設備不堪負荷下放棄使用此方法，改為原本的只輸入原 mfcc。

- Word2vector

  使用 facebookresearch 所提供的中文 wordembeding，將每個中文單字以 300 維表示。

  資料中每個句子最長為 13 個單字，所以我們將每個句子轉為 300x13 的 vector，不足 13 個字的句子就把後面補 0。

- Fake data

  我們將 input 資料 mfcc 及對應的正確答案 w2v 當作真資料，並另外人為的產生假資料讓 model 能更好的判斷真偽，以下是幾種我們嘗試的製造假資料的方法。

  法 1：讓第 n 個 mfcc 檔案與第 n+1 個 w2v 做對應

  優點： 每個 w2v 的句子皆為真句子。

  缺點： 因為每個句子的長短不同，model 可能透過句子長短來判斷真偽，而在測試的 4 個選項中每個句子的長短是一樣的。

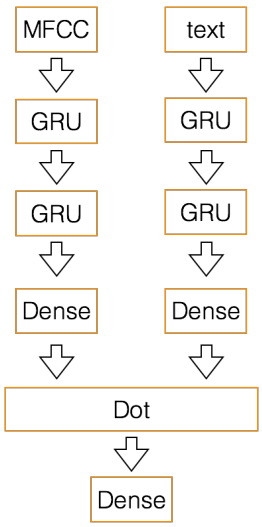
  法 2：將每個在 training set 中的中文字加入字典中，根據每筆資料原本的句子長度，從字典中隨機選字，產生相對的 w2v。

  優點： 真假資料句子長度相同，model 必須從 w2v 的差異上分辨出真假資料的差異。

  缺點： 假資料為中的文字為隨機填上，model 可能可以從句子是否通順來判斷真偽，而測試的 4 個選項中每個句子皆為通順的。

■ 使用模型（Best Model）：

- Retrieval model
  - 模型架構：



- 模型詳細參數(best)如下：

```
def gen_model_pretrain_GRU():
    mfcc_input = Input(shape=[246, 39])
    text_input = Input(shape=[13, 300])

    hidden = GRU(256, return_sequences=True, dropout=0.3,
                 recurrent_dropout=0.3)(mfcc_input)
    # hidden = LSTM(256, return_sequences=True, dropout=0.2,
    #               recurrent_dropout=0.2)(hidden)
    hidden = GRU(256, dropout=0.3, recurrent_dropout=0.3)(hidden)
    mfcc_output = Dense(512)(hidden)

    hidden = GRU(256, return_sequences=True, dropout=0.3,
                 recurrent_dropout=0.3)(text_input)
    # hidden = LSTM(256, return_sequences=True, dropout=0.2,
    #               recurrent_dropout=0.2)(hidden)
    hidden = GRU(256, dropout=0.3, recurrent_dropout=0.3)(hidden)
    text_output = Dense(512)(hidden)

    r_hat = Dot(axes=1)([mfcc_output, text_output])  # merge layer
    # r_hat = concatenate([mfcc_output, text_output])

    output = Dense(1, activation='sigmoid')(r_hat)

    model = Model([mfcc_input, text_input], output)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam', metrics=['acc'])
    model.summary()
```

- 模型想法說明：

  我們將此問題轉化成一個 regression 的問題，將正確的 MFCC 音訊檔對應 Text 文字標記為 1，將人為產生的假資料(使用上述假資料做法 1)給予標籤 0。

  模型架構很簡單，基本上就是將 MFCC 音訊檔與 Text 文字分別丟進上述架構中的 RNN 模型，再來將這兩個 RNN 模型的輸出分別送進一層的 NN，接著兩個 NN 的結果再 Dot 起來，最後再經過一個 sigmoid 的 activation function 來獲得 model 所預測的標籤。

  因為這邊已經轉化成簡單的二元分類問題，因此我的 loss function 選用 binary cross entropy。

- 實驗設置及結果

  實驗 1：基於上述的 RNN 架構下，使用不同的參數進行實驗，結果如下表。

RNN cell	#of layers	#of nodes	dropout	Recurrent_dropout	score
LSTM	1	(128)	(0,0)	(0,0)	0.331
LSTM	1	(256)	(0,0)	(0,0)	0.337
LSTM	2	(256,256)	(0,0)	(0,0)	0.46
LSTM	2	(256,256)	(0.3,0.3)	(0.3,0.3)	0.55
GRU	2	(256,256)	(0.3,0.3)	(0.3,0.3)	0.60

可以觀察得到，只要 LSTM 疊越多層，score 就會有明顯的上升，如再加上 dropout 及 recurrent dropout 更上升了不少，因此可以推斷出在沒有實作 dropout 的結果很有可能 overfitting 了。

相同的模型架構把 LSTM 改成 GRU，score 直接上升了 0.05，猜測是因為 GRU 使用的參數比 LSTM 更少，所以更加不會有 overfitting 的現象。

實驗 2:使用上述表現最好的 GRU 來實現模型，並且比較不同的假資料比例會不會影響 score 的表現：

真:假	Score
1:1	0.54
1:2	0.58
1:3	0.6

可以推測出相同的模型架構下，1:1 的的資料分配是不佳的，原因可能是機器分辨不出來哪一個標籤比較具有分辨的價值，也就是判斷 1 比判斷 0 還要來得有價值。

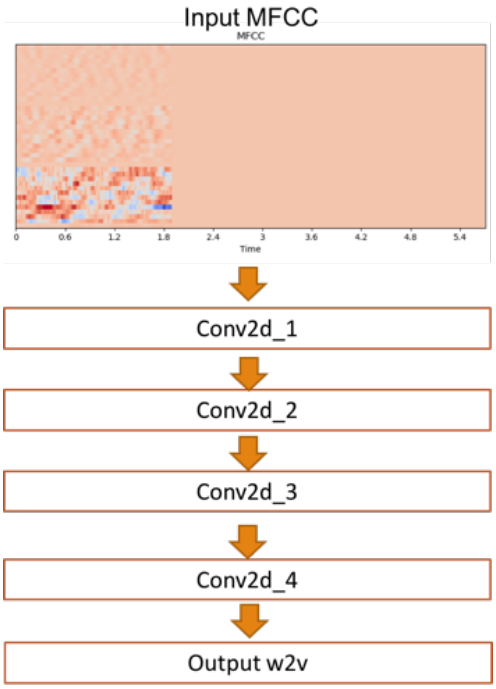
我們 model 可以藉由眾多假資料中學習到真假資料的差別，進而讓 model 更容易預測出正確的資料。當然可以想像如果假資料太多，那就會形成 data unbalance 的問題，就目前所言，1:3 是一個不錯比例，在 Score 上表現最佳，而且也較符合測試時的環境（4 選 1）

- 嘗試過的模型：
  - CNN model
    - 想法說明：

      mfcc 的 input 可以為二維的矩陣，並且可以視覺化為圖形表示（上述資料前處理提及）。

      而 w2v 同時也是二維矩陣，所以我們試圖使用辨識圖形效果顯著的 CNN 模型，希望將 input mfcc 檔案經過 CNN 之後，轉換為其對應的 w2v，試圖實現透過輸入音訊後直接預測出對應的中文句子。

- 模型架構：



詳細參數如下：

layer	Kernel_num	Kernel_size	Stride_size	Function
Conv2d_1	128	(13,39)	(3,1)	Selu
Conv2d_2	512	(10,1)	(3,1)	Selu
Conv2d_3	512	(7,1)	(1,1)	Selu
Conv2d_4	300	(5,1)	(1,1)	Selu

- 模型說明：

使用 4 層的 CNN 架構，並且在每層 CNN 之前加上一層 batch_normlize，調整 kernel_szie 以及 Stride_size 讓 model 的 ouput_size 與目標 w2v 相同。在最後一層 Conv2d_4 中，把 Kernel_num 調整為 300 並且將這個參數視為 word embedding（上述資料前處理中提及）中表示每個中文字的 300 維。

optimizer：SGD　使用 learning rate：0.01

loss function：triplet_margin_loss (pred,true,false)

在減少 pred 與 true data 的 L1 distance 的同時，也要增加與 fake data 的 L1 distance，並給予一 margin 值避免 model 過度增加與 fake data 的 L1 distance。詳見：

http://pytorch.org/docs/0.3.0/nn.html?highlight=triplet_margin_loss#torch.nn.functional.triplet_margin_loss

- 實驗設置：

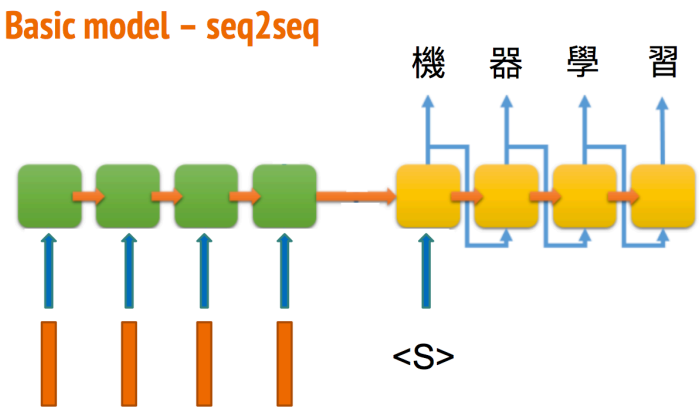
  使用 40000 筆 data 作為 training set，剩餘 5036 筆作為 val set。fake data 作法採用上述法 2，在 val set 上的每筆資料使用 3 個假資料與原本真資料補成 4 個選項，來模擬在 test 上的環境。

  讓此模型在這 4 個選項與 model output 計算 L1 distance，取最小值當作答案。

  Model 在 val set 上有接近 55%準確率的表現，但在 kaggle 上最高成績不如上述 Retrival model，之後放棄此架構，改在 Retrival model 上進行更多實驗及改進。

➢ Sequence to sequence

- 模型架構：



- 模型詳細參數如下：

```
def seq2seq(hidden_dim, output_length, output_dim):
    model = Sequential()
    model.add(Bidirectional(LSTM(128, return_sequences=False),
                            input_shape=(246, 39)))
    model.add(Dense(64, activation="relu"))
    model.add(RepeatVector(13))
    model.add(Bidirectional(LSTM(128, return_sequences=True)))
    model.add(TimeDistributed(Dense(output_dim=300, activation="linear")))
    model.compile(loss='mse', optimizer='adam')

    model.summary()
```

- 模型想法說明：

  這是我第一個實踐的的模型，想法是把 MFCC 音訊檔丟到 RNN 中，然後輸出一個[13, 300]大小的 vector，但是不管我怎麼調整，辨識率都是可憐的 20 幾趴，連 simple baseline 都突破不了，所以在逼近死線之際，我放棄了這個做法，改為下述的 retrieval model。

- 實驗設置及結果

  中途有嘗試實驗 LSTM 跟 Bidirectional 的差別，結果如下：

RNN cell	Score
LSTM	23.9
Bidirectional LSTM	26.1

  可以觀察得到 Bidirectional LSTM 表現的比較好，但由於分數實在太低，故無法做出什麼推斷。

■ 總結及討論：

- 原始的語音資料有些背景雜音且每句話語速不固定，再加上是要從台語語音轉換為中文字，這個目標較為困難。我們在 Retrival model 中將問題轉換為一個 regression 的問題，讓 model 只需判斷真偽，藉此簡化問題。
- 我們嘗試使用了基於 CNN 以及基於 RNN 來實作的 2 種模型，而結果上 RNN 模型有較好的效果，可以推測對於語音資料方面，時間序列上的特徵會成為模型判斷的一個重要依據。
- 在 RNN 的模型中，GRU 的表現普遍來說比 LSTM 還要來得好，推測可能是在這個問題上參數的多寡會影響結果是否 overfitting
- 除了原本所提供的正確資料，人為產生假資料也可以很大的輔助模型做出正確的預測。在這之上，如何產生假資料與真假資料的比例也值得考究。一開始我們沒有仔細定義假資料的做法，讓模型完全訓練不起來。