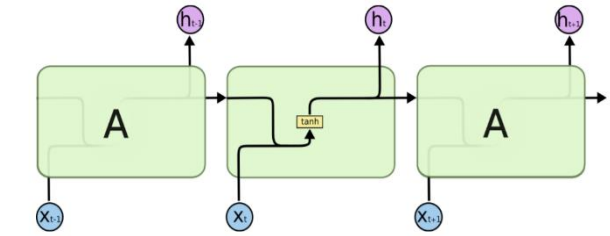
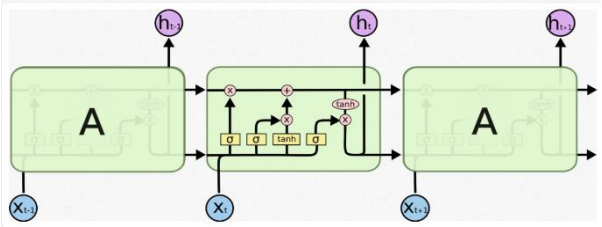
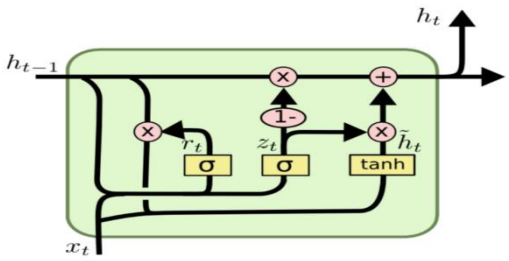
[ADL 作業一 report]

- Model description
  - RNN:

    RNN 和 NN 不同的地方在於多了時間的資訊，所以適合用來預測 sequential information，即本次作業的 phone prediction。在 keras 中提供了三種 RNN，SimpleRNN、LSTM、GRU，SimpleRNN 是最標準一般的 RNN，每個神經元除了現在這個時間點的輸入值，還會有前一時間點傳來的資訊，幫助 RNN 記住之前發生過的事以做出更準確的預測，然而這常會遭遇到梯度消失、梯度爆炸的問題，所以就又發明出 GRU 和 LSTM，兩者主要的概念都是新增閘門來控制某些資訊是否可以往下傳。LSTM 除了原本貫穿 LSTM 的 cell state 外，多了三個控制器，input gate、forget gate 和 output gate，input gate 是按此時輸入對結果的重要性來按比例傳入輸入值，某些對結果影響不大的輸入就可能被忽略，forget gate 同樣依重要性來按比例決定不傳入之前的某些資訊，替換成新的資訊，最後 output gate 再控制在這個時間點下，那些資訊需要輸出，所以如果把輸入控制設為 1，忘記控制為 0，輸出控制為 1，LSTM 就換變成 simpleRNN。GRU 可以看成是 LSTM 的變形，把 LSTM 中的 forget gate 和 input gate 用 update gate 來代替，也把 cell state 和 h_t 合併。所以只有兩個控制器，reset gate 控制需保留多少之前的記憶，如果為零，就只包含當前輸入的訊息，update gate 控制需忘記多少前層記憶，加上新的輸入訊息，直接成為當層輸出的結果，所以需要長時間記憶時 update gate 值會比較大，反之，就會是 reset gate。







圖上:simpleRNN
圖中:LSTM
圖下:GRU

  - RNN+CNN

    CNN 是透過一個 filter 來做卷積，每次掃描一小區域進行學習，所以適合用在圖像辨識上，能利用到相鄰位置的資訊，在這次作業中我使用一維的時域卷積，即 keras 中的 Conv1D，他只對寬度進行卷積，而不會對深度也進行卷積

- How to improve your performances

  最一開始我並未對輸入做太多前處理，只是單純一個個 frame 讀進來，並預測，如此用簡單的 NN 就有四成多的準確率，但不管如何調參數都無法再提升，後來改用 RNN，使用 SimpleRNN，結果和 NN 時差不多，也是四成多準確率，因為我並沒有利用到 RNN 對 sequence information 的特性，所以我開始嘗試其他前處理。

  首先是把 frame 組合成一個句子，接著做 padding，我嘗試過全部補零，用 sil 來補不足部分，但因為每個句子長度差很多，導致補的零太多，雖然可有五六成的準確率，但實際 test 的結果就很差，因此改為做 repeat，即重複句子來補不足部分，如此雖然可以做到七成七的準確率，但實際 test 效果也是不佳，分數 30.40，所以最後改為使用 window 的概念，不針對句子做 padding，而是每個 frame 取前 15 個和後 15 個來補成總長 31 的 window，如果不夠則是採取補零方式，也就是在預測每個 frame 時給 RNN 前後 frame 的 hint，如此也差不多可以到七成七的準確率，但實際 test 的結果比前一種方法好，只是仍然和 baseline 差 0.17。

  在這次作業中，我使用了 LSTM，經過嘗試，LSTM 和 GRU 似乎差異不大，所以就還是選擇前者。另外還使用了 BatchNormalization 將每一層結果做標準化，依據網路上找到的資料，因為訓練時每一層輸入的分布在變化，導致訓練過程中的飽和，稱這種現象為：internal covariate shift，需要降低學習率和注意參數的初始化，而 google 提出的 BatchNormalization 方法是對每一個 batch 都進行標準化(正態化)，如此也不需要使用 Dropout 方法避免 overfit，收斂速度也會比較快。而一開始使用簡單 NN 時，每一層的 cell size 我只用 50，但隨著不同 model 的測試，數字一再加大，最後用到 512。至於 CNN model，我是在原本 RNN model 上再疊一層 Conv1D，因為我們並不是要辨識圖片，而 Conv1D 只對寬度進行卷積，而不會對深度也進行卷積，也能利用到相鄰時間序列的關係，但實際執行結果並沒有比單純 RNN 好，準確率只有七成出頭，分數 20 幾分，所以最後 best model 採用原本的 RNN model。