HW1

1. Model description

RNN:

下圖為 model 架構，只使用 MFCC（39 維），

```
model = Sequential()
model.add(BatchNormalization(input_shape=(777,39)))
model.add(Bidirectional(GRU(200,implementation=2,return_sequences=True,activation='tanh',
                            recurrent_activation='sigmoid',dropout=0.2)))
model.add(Dense(100,activation="selu"))
model.add(Dropout(0.1))
model.add(Dense(40,activation="softmax"))
model.compile(loss='categorical_crossentropy',optimizer="adam", metrics=['accuracy'])
model.summary()
```

第一層先用 batchnormalization layer 將 batch data 做 normalize，希望可加速收斂與降低 overfitting 之可能，再來直接輸入 rnn layer，rnn 採用雙向的 GRU 以更少的參數達到和 LSTM 相同效能，implementation 設為 2 可以讓 GRU 能更有效率的執行，而輸出也是 sequencial 的輸出，這邊輸出為 200 維，但因為雙向的關係，所以總共會有 400 維，採用雙向是因為連同 backward 的 sequences 都考慮，最後接上兩層 dense，第一層的 activation 為 selu，他可以達到 batch normalization 的效果，第二層為輸出 40 維並使用 softmax 使 output 為 40 個 classes 的 probabilities，之所以為 40 個 class，是因為已經先將 48 個 phones 先轉乘 39 個 phones，再多加一種 phone 作為 input data sequence 的 padding 補零之對應。

RNN+CNN :

下圖為 model 架構，只使用 MFCC（39 維），

```
model = Sequential()
model.add(GaussianNoise(0.3,input_shape=(777,39)))
model.add(Convolution1D(200,4,padding='causal',dilation_rate=1, activation='selu'))
model.add(Dropout(0.3))
model.add(Convolution1D(150,4,padding='causal',dilation_rate=2, activation='selu'))
model.add(Dropout(0.3))
model.add(Convolution1D(120,3,padding='causal',dilation_rate=4, activation='selu'))
model.add(MaxPooling1D(3))

model.add(Convolution1D(80,2,padding='causal', activation='selu'))
model.add(Dropout(0.3))

model.add(Bidirectional(GRU(200,implementation=2,return_sequences=True,
                            activation='tanh',recurrent_activation='sigmoid',recurrent_dropout=0.2,dropout=0.3)))
model.add(Bidirectional(GRU(200,implementation=2,return_sequences=True,
                            activation='tanh',recurrent_activation='sigmoid',recurrent_dropout=0.2,dropout=0.3)))
model.add(Dense(150,activation="selu"))
model.add(UpSampling1D(3))
model.add(Dropout(0.3))
model.add(Dense(100,activation="selu"))
model.add(Dropout(0.3))
model.add(Dense(40,activation="softmax"))
model.compile(loss='categorical_crossentropy',optimizer="adam", metrics=['accuracy'])
model.summary()
```

第一層以 GaussianNoise layer 取代 batchnormalization layer，將 input signal 加入點雜訊可以讓 training 更有效率，也可以降低 overfitting，再接上 3 層的 convolution1D layers，為只針對 frame 的維度進行 convolution，並採用 causal padding 並搭配 dilation rate，這是參考 google wavenet 的 convolution layers，該 model 也是對原始語音 data 做 training，causal convolution 使 t 時刻的輸出僅僅依賴於 t 以前的輸入，不會依賴於 t+1 時刻以及之後的輸入，而 dilated 的好處是不做 pooling 損失訊息下加大感知範圍讓每個 convolution 輸出都包含較大範圍的訊息，三層 convolution 後再做 maxpooling，之後再做一次 convolution，這邊所有 convolution 都是採用 selu activation 也是希望達到 batchnormalization 的效果，再來接上兩層的雙向 GRU，有測過確實兩層比一層準確率更高。做完 RNN 後再加上 Dense layer，再做 upsampling 將 output 的 frame 數放大三倍，變為原來的 frame 數，在接上兩層 Dense，並輸出 40 個 class 的 probabilities。

2. How to improve your performance

於 CNN+RNN model，提到說第一層採用 GuassianNoise 取代 normalization，除了可以降低 overfitting，提高 generalization，是參考 paper (Phoneme Recognition in Timit with BLSTM-CTC)，也比使用 batchnormalization 的 model 於 test data 的準確度高出一些。

這邊大部份的 layer 都使用 selu activation，以下是 selu 的公式，

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

其中
λ=1.0507009873554804934193349852946, α= 1.67326324235437728
48170429916717，兩個神奇的參數是 paper(Self-Normalizing Neural Network)作者用各種數學證明算出來的(主要是 Banach Fix Point Theorem)，可以將 input 自動做 normalization。

3. Experimental results and settings

Compare and analyze the results between RNN and CNN

在 RNN layers 前加上 CNN layers 與 maxpooling 可以先將 input 做 features extraction，讓餵給 RNN 的 input 較會是重點的 features，使得 RNN 在 train sequence to sequence 時不會因為 confuse，而這邊採用的是 convolution1D，為針對 time sequence (frame)做 convolution，依照 kernel size 來決定要感知多少的 sequences。

可以從 model predict 的結果來看，只有 RNN 的話 predict 的 phones 會很容易在連續相同的 phones 中間出現不同的 phones，但 ground truth 往往是一段一段連續的 phones，這會使得 predict 跟 ground truth 的 edit distance 頗高，但由於在 training 時，我們 loss function 是針對 frame wise 的 predict 的 phone 去做計算，不會去考慮到所有 frame 的結果。 所以我再 RNN layers 前面加了 CNN 與 maxpooing，然後將 RNN 後面再做 upsampling，如此一來連續 output 的 phones 會是 3 個以上，減緩了上述之情形，而 edit distance 也下降許多。

Compare and analyze the results with other models

有嘗試不同的 RNN 的準確度，發現同樣參數設定的 GRU 與 LSTM，他們的成效是差不多的，而 GRU 所需要 update 的參數反而比 LSTM 少許多，training 的速度也較快，所以最後還是使用 GRU 作為 RNN layer。