IIE Individual Research Project

以深度學習方法實作簡單語音辨識模型

學生：107034703 陳家安

指導老師：邱銘傳 教授

摘要

有鑑於近年來人工智慧在各領域的應用大幅增加，了解深度學習的基本概念及程式上的實作也成為重要學習目標，能夠根據所得到的原始資料，進一步去分析、了解資料的特性，再去選擇及使用各種不同的神經網路來建構模型更是一個重要的課題。

在深度學習的應用領域中，最大宗的莫過於圖像辨識（Image Recognition）及自然語言處理（Natural Language Processing），因此本研究想針對後者的領域進行一簡單的程式實作，將根據課堂所學以及網路上所閱讀之多元教材，利用卷積神經網路(Convolutional Neural Network，CNN)及長短期記憶模型（Long Short-Term Memory，LSTM）實作一簡易語音辨識模型，希望能對簡單的單詞進行辨識，也透過調參設計及實驗，以期發展一高準確率的辨識模型。

關鍵字：深度學習、神經網路、語音辨識、卷積神經網路(CNN)、長短期記憶模型(LSTM)

目錄

一、 研究動機與目的

有鑑於平時上課或聽演講常使用到即時語音辨識並轉文字的 APP，對人聲等音頻檔案的辨識方式有興趣，因此想藉此專題的實作機會，更了解其中邏輯，並針對一些單詞進行簡單的辨識模型建構。本研究將選擇一公開語音資料集—Google 的 Speech Commands Dataset，並透過資料前處理過程，利用 Python 的 scipy.io 套件的 wavfile 將音檔的頻率視覺化變成圖片，再透過深度學習的模型進行訓練，使其能夠辨識出一些簡單的單詞，並透過調整參數提升準確率。

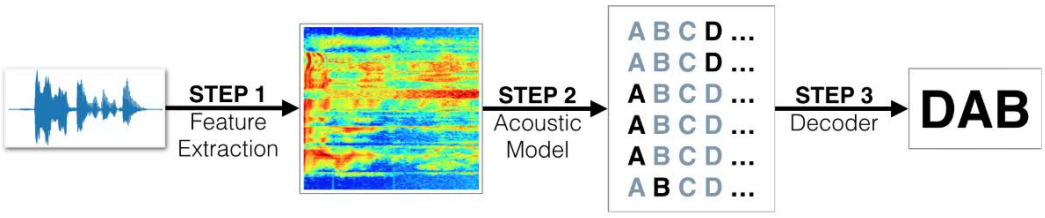
二、 文獻探討

1. 語音辨識(Speech to text)

語音辨識為電腦透過比對聲學特徵，進而將說話者的發音內容轉化為文字的技術。此領域在 1980 年代就由美國麻省理工學院的實驗室發起研究，但苦於辨識率不高，一直沒辦法應用在商業用途。直到 2012 年，科學家用深度神經網路(Deep neural network，DNN)的計算方式取代傳統高斯分配計算，使辨識率大幅提升，才逐漸受到國際間大型企業的關注與重視。

2. 應用深度學習模型於語音辨識

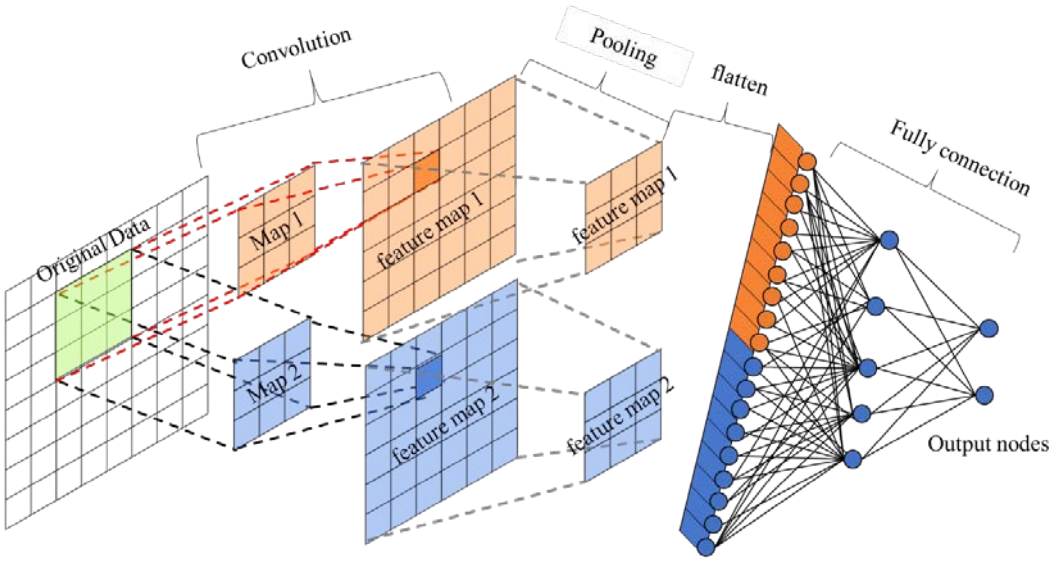
使用深度網路實現自動語音辨識(Auto Speech Recognition, ASR)的主要流程為：輸入語音片段(Spectrogram、MFCCs……等) ，將原始語言轉換為聲學特徵，再經過神經網路的判斷及機率分布，最後輸出對應的文本內容。



圖一、深度網路實現自動語音辨識流程

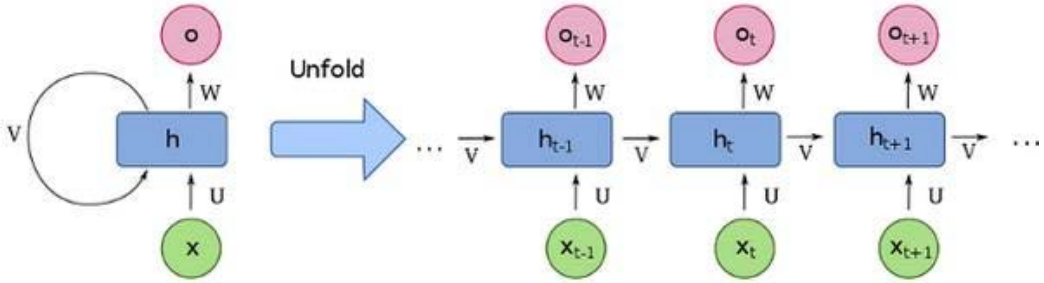
其中本研究使用的兩神經網路為卷積神經網路(Convolutional Neural Network，CNN)及遞歸神經網路(Recurrent Neural Networks，RNN)的長短期記憶模型（Long Short-Term Memory，LSTM）。

CNN 為卷積神經網路由卷積層、全連通層、池化層組成，搭配反向傳播演算法的運算，能夠利用輸入資料的二維結構擷取特徵並適當的收斂與學習，在圖像和語音辨識方面有出色的表現，其架構如下圖所示。



圖二、CNN 架構

RNN 則是一個擁有稱為 LSTM 這種活性資料記憶體的神經網路，可以用於一系列資料以猜測接下來會發生的內容，其輸出不僅與當前輸入和網路的權值有關，也與之前網路的輸入有關，常用在處理時序資料。現在已大量運用在自然語言理解 (例如語音轉文字，翻譯，產生手寫文字)，圖像與影像辨識等領域，其運算概念如下圖所示。



圖三、RNN 運算概念

三、 研究方法

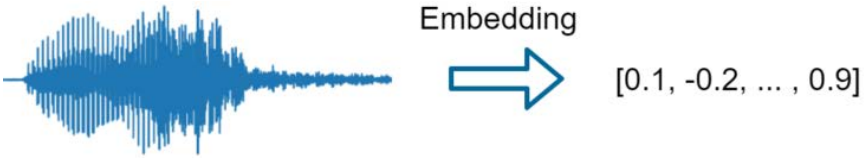
本研究使用 Google 的公開訓練資料集—Speech Commands Dataset 進行分析與深度學習模型訓練，內含 30 種不同單詞音檔，每個詞都有約 2300~2400 個原始 wav 音檔。將以此資料集為基礎，進行資料前處理(包含聲波的分析及轉換、訓練及測試資料的切割……等)，並於 python 環境中利用 keras 及多種套件，建構卷積神經網路模型以及長期短期記憶模型，對轉換後資料的進行圖片辨識的訓練。

四、　個案研究與實作

1. 資料前處理

本研究從資料集共 30 個不同單詞中，選擇三個來進行模型的訓練，這三個單詞分別為：「yes」、「no」和「up」(分別共有 2377、2375、2375 筆資料)，並將原始音頻檔依照下列手法進行資料前處理：

(1) 用 scipy 的 wavfile.read 讀取 wav 檔，並將字串用 array 的方式呈現 (Embedding)，並進行正規化



圖四、Embedding 概念

(2) 將音檔與依照給定的 sample rate 進行比較，長度太長的部分隨機截斷，若長度太短則隨機加上 silence 的部分。

```
# 如果長度太長則隨機截斷
if len(wav) > L:
    i = np.random.randint(0, len(wav) - L)
    wav = wav[i:(i+L)]
# 若太短則隨機加上silence
elif len(wav) < L:
    rem_len = L - len(wav)
    silence_part = np.random.randint(-100,100,16000).astype(np.float32) / np.iinfo(np.int16).max
    j = np.random.randint(0, rem_len)
    silence_part_left  = silence_part[0:j]
    silence_part_right = silence_part[j:rem_len]
    wav = np.concatenate([silence_part_left, wav, silence_part_right])
```

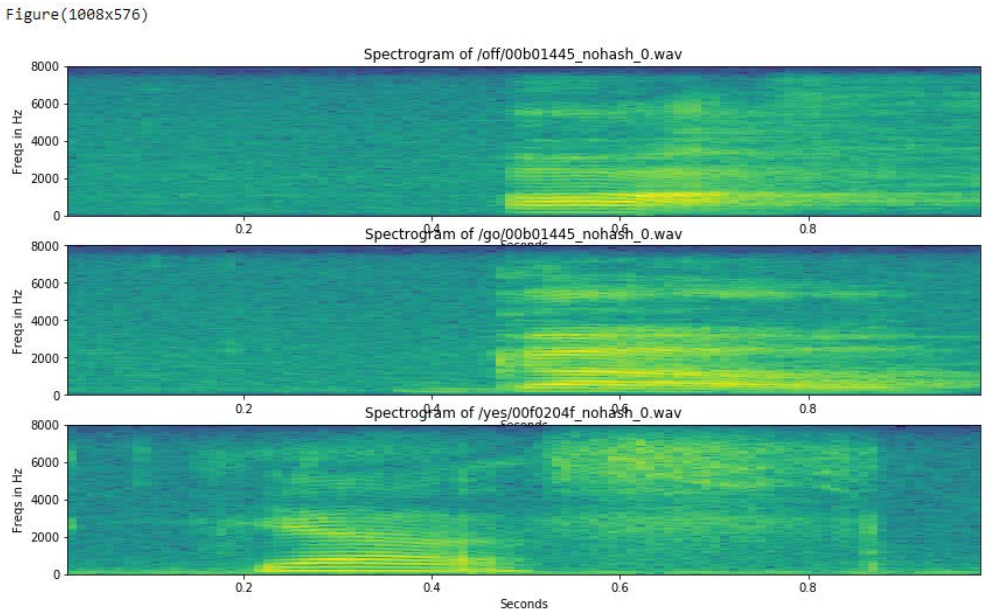
其中取樣頻率（sample Rate）為每秒鐘所取得的聲音資料點數，以 Hertz（簡寫 Hz）為單位，點數越高，聲音品質越好，但是資料量越大。例如 24kHz 等於每秒鐘對這個聲音切 24000 次，

間斷性的採樣可能跟原始聲波有一定差距，可以根據採樣定理(Nyquist theorem)，利用數學從間隔的採樣中完美的重建原始模擬聲波，只要以我們希望得到的最高頻率的兩倍來採樣即可。本研究中將 sample Rate 設為 16000，故頻率區間為[0, 8000]。

此外，人類聽覺範圍是 20Hz 至 20000Hz。

(3) 將聲波用 Short-time Fourier Transform(STFT)的方法轉換成時頻譜 (spectrogram)，其中時頻譜為一種描述波動的各頻率成分如何隨時間變化的熱圖，其 y 軸為微小時間間隔內聲波的頻率分布，x 軸為各頻率隨時間的變化

```
freqs, times, spec = stft(wav, L, nperseg = 400, noverlap = 240, nfft = 512, padded = False, boundary = None)
```



圖五、時頻譜範例

(4) 去除大於 threshold 頻率的部分，在此設為 5500 赫茲

(5) 將三種單詞的資料以 5:2:3 的比例隨機拆分為 training、validation 及 testing 三部分，其中訓練集用來訓練模型的引數、驗證集用來確定網路結構及協助人工調整參數，會在每個 epoch 完成後用來測試一下當前模型的準確率。測試集則用於驗證模型最終性能。

```
# 拆分測試資料
def apply_train_test_split(self, test_size, random_state):
    self.df_train, self.df_test = train_test_split(self.df,
                                                   test_size=test_size,
                                                   random_state=random_state)
# 拆分驗證資料
def apply_train_val_split(self, val_size, random_state):
    self.df_train, self.df_val = train_test_split(self.df_train,
                                                  test_size=val_size,
                                                  random_state=random_state)
```

```
# 將資料集分為training, validation and testing part
dsGen.apply_train_test_split(test_size=0.3, random_state=2019)
dsGen.apply_train_val_split(val_size=0.2, random_state=2019)
```

2. 模型說明

本研究兩種模型來進行模型的訓練，包含擅長處理圖像的 CNN 及擁有記憶能力的 RNN(LSTM)，兩模型主要架構如下：

(1) CNN

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNormalization

def deep_cnn(features_shape, num_classes, act='relu'):

    x = Input(name='inputs', shape=features_shape, dtype='float32')
    y = x

    # Block 1
    y = Conv2D(32, (3, 3), activation=act, padding='same', strides=1, name='block1_conv', input_shape=features_shape)(y)
    y = MaxPooling2D((3, 3), strides=(2,2), padding='same', name='block1_pool')(y)
    y = BatchNormalization(name='block1_norm')(y)

    # Block 2
    y = Conv2D(32, (3, 3), activation=act, padding='same', strides=1, name='block2_conv')(y)
    y = MaxPooling2D((3, 3), strides=(2,2), padding='same', name='block2_pool')(y)
    y = BatchNormalization(name='block2_norm')(y)

    # Block 3
    y = Conv2D(32, (3, 3), activation=act, padding='same', strides=1, name='block3_conv')(y)
    y = MaxPooling2D((3, 3), strides=(2,2), padding='same', name='block3_pool')(y)
    y = BatchNormalization(name='block3_norm')(y)

    # Flatten
    y = Flatten(name='flatten')(y)

    # Dense layer
    y = Dense(64, activation=act, name='dense')(y)
    y = BatchNormalization(name='dense_norm')(y)
    y = Dropout(0.2, name='dropout')(y)

    # Predictions
    y = Dense(num_classes, activation='softmax', name='pred')(y)

    # Print network summary
    Model(inputs=x, outputs=y).summary()

    return Model(inputs=x, outputs=y)
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 177, 98, 1)]      0
_________________________________________________________________
block1_conv (Conv2D)         (None, 177, 98, 32)       320
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 89, 49, 32)        0
_________________________________________________________________
block1_norm (BatchNormalizat (None, 89, 49, 32)        128
_________________________________________________________________
block2_conv (Conv2D)         (None, 89, 49, 32)        9248
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 45, 25, 32)        0
_________________________________________________________________
block2_norm (BatchNormalizat (None, 45, 25, 32)        128
_________________________________________________________________
block3_conv (Conv2D)         (None, 45, 25, 32)        9248
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 23, 13, 32)        0
_________________________________________________________________
block3_norm (BatchNormalizat (None, 23, 13, 32)        128
_________________________________________________________________
flatten (Flatten)            (None, 9568)              0
_________________________________________________________________
dense (Dense)                (None, 64)                612416
_________________________________________________________________
dense_norm (BatchNormalizati (None, 64)                256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
pred (Dense)                 (None, 3)                 195
=================================================================
Total params: 632,067
Trainable params: 631,747
Non-trainable params: 320
_________________________________________________________________
```

包含了三個卷積層、三個最大池化層、一個扁平層及兩個全連接層。

```
model = deep_cnn(INPUT_SHAPE, NUM_CLASSES)
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['acc'])
```

其中在模型訓練過程中也加入了一個提早暫停的機制(callbacks)，當發現驗證準確率開始有下降趨勢後，模型便會提早結束訓練。

```
callbacks = [EarlyStopping(monitor='val_acc', patience=4, verbose=1, mode='max')]

history = model.fit_generator(generator=dsGen.generator(BATCH, mode='train'),
                              steps_per_epoch=int(np.ceil(len(dsGen.df_train)/BATCH)),
                              epochs=EPOCHS,
                              verbose=1,
                              callbacks=callbacks,
                              validation_data=dsGen.generator(BATCH, mode='val'),
                              validation_steps=int(np.ceil(len(dsGen.df_val)/BATCH)))
```

(2) RNN(LSTM)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.layers import LSTM

model = Sequential()
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
callbacks = [EarlyStopping(monitor='val_accuracy', patience=4, verbose=1, mode='max')]
history = model.fit_generator(generator=dsGen.generator(BATCH, mode='train'),
                              steps_per_epoch=int(np.ceil(len(dsGen.df_train)/BATCH)),
                              epochs=EPOCHS,
                              verbose=1,
                              callbacks=callbacks,
                              validation_data=dsGen.generator(BATCH, mode='val'),
                              validation_steps=int(np.ceil(len(dsGen.df_val)/BATCH)))
```

3. 參數調整與訓練成果

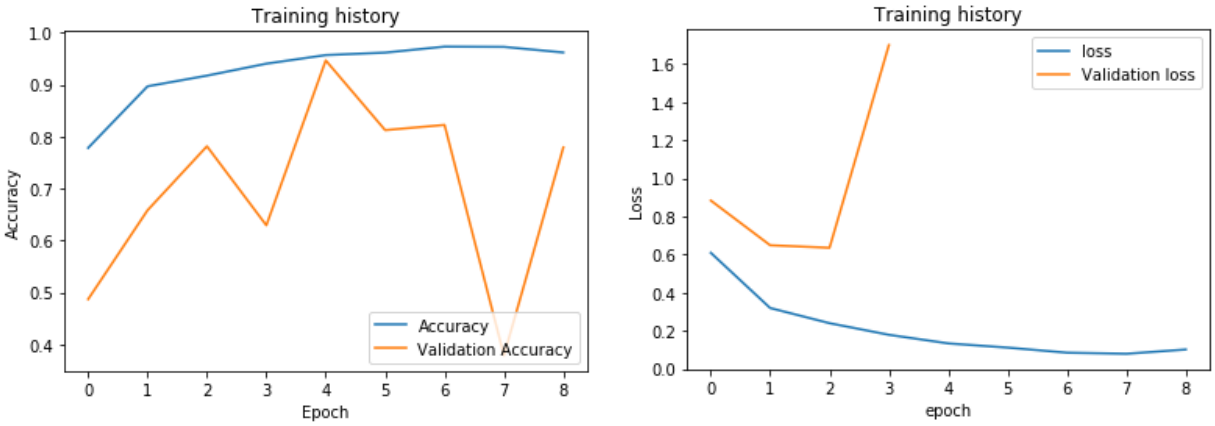
(1) CNN 模型：

此部分主要針對 epoch、batch size、CNN 的激活函數及 Optimizer 進行調參來優化模型及準確率，共進行以下 6 個實驗：

I. 實驗一：

- Epoch：15
- batch size：32
- CNN 激活函數：relu / 最後一層全連接層使用 sigmoid

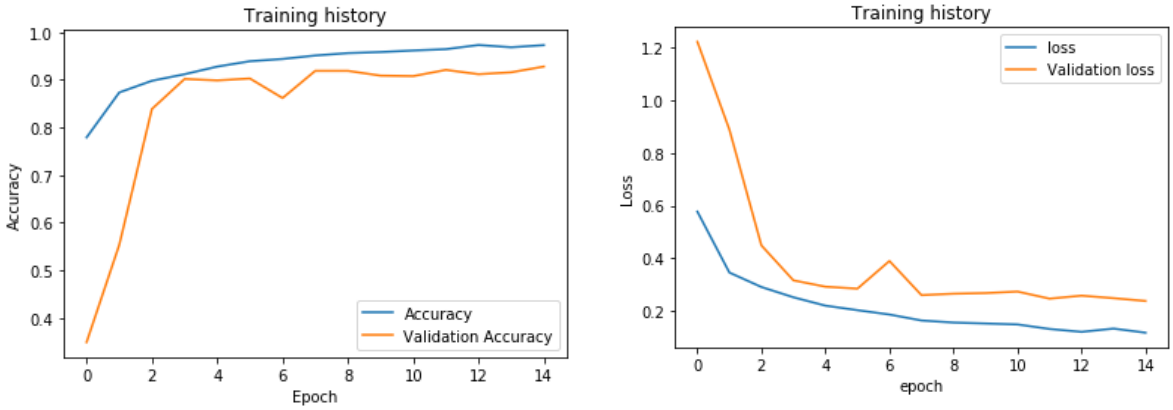
- Optimizer：Adam



共 125 個 batch 進行訓練，在第 9 個 epoch 時提早停止，最終模型準確率為 77.8%。

II. 實驗二：

- Epoch：15
- batch size：32
- CNN 激活函數：relu / 最後一層全連接層使用 softmax
- Optimizer：Adagrad

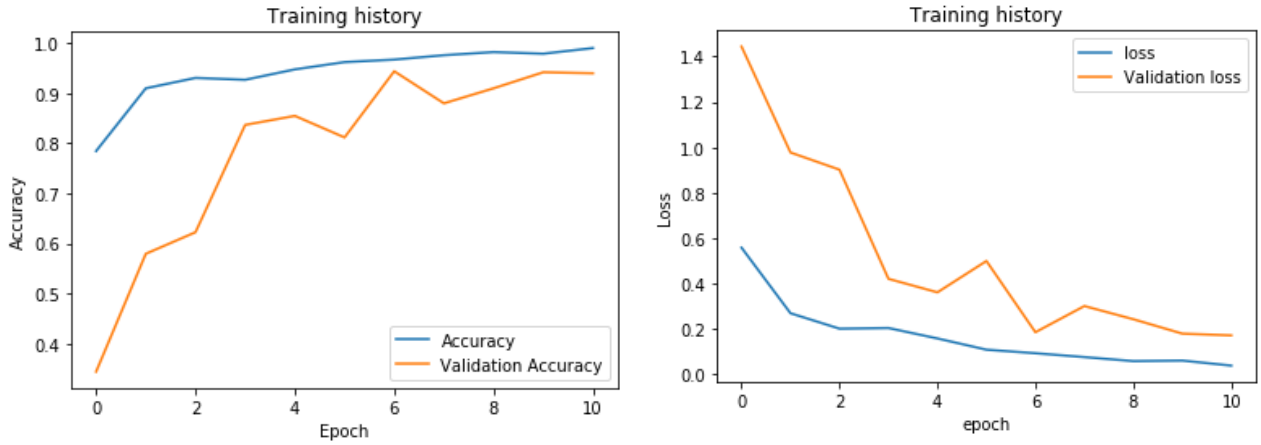


共 125 個 batch 進行訓練，最終測試準確率為 92.0%。

III. 實驗三：

- Epoch：15
- batch size：32
- CNN 激活函數：relu / 最後一層全連接層使用 softmax

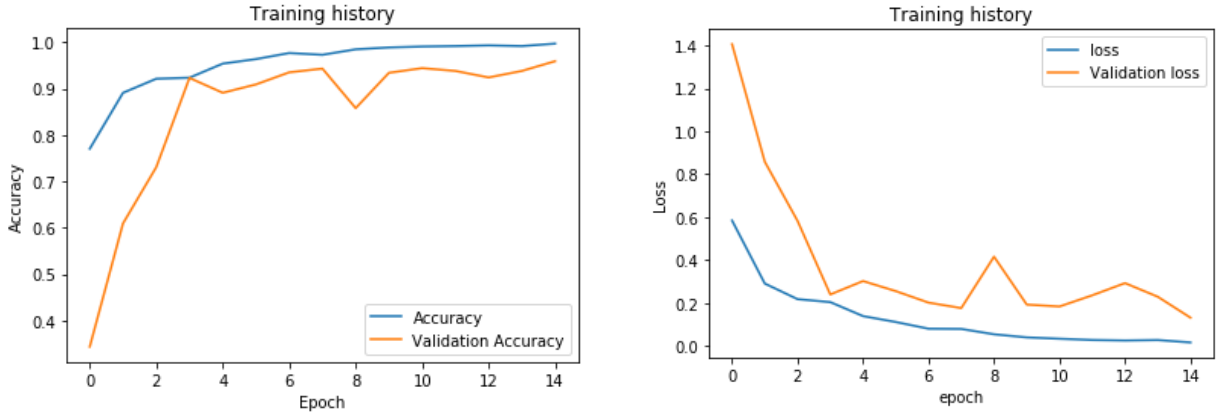
- Optimizer：Adam



共 125 個 batch 進行訓練，在第 11 個 epoch 時提早停止，最終模型準確率提升到 94.4%。

IV. 實驗四：

- Epoch：15
- batch size：64
- CNN 激活函數：relu / 最後一層全連接層使用 softmax
- Optimizer：Adam

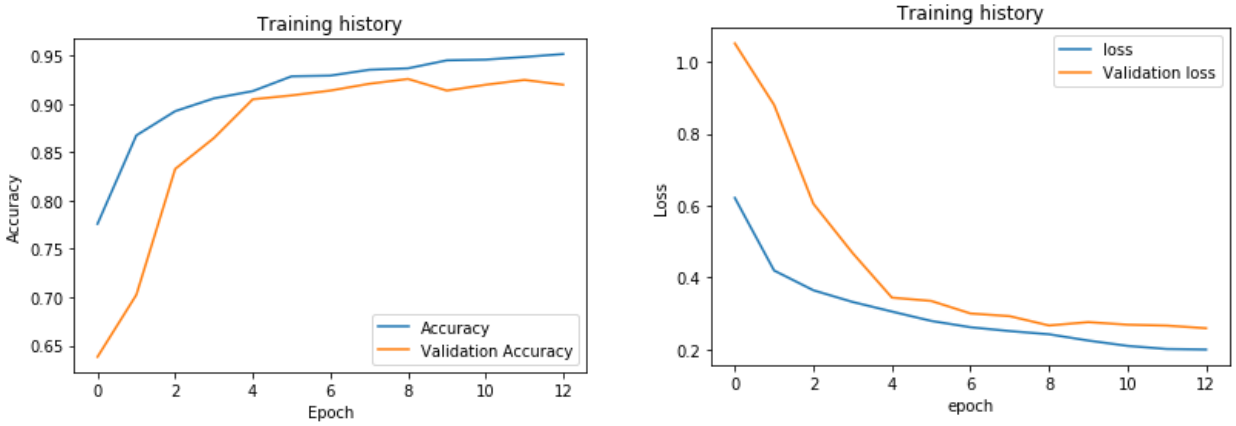


共 63 個 batch 進行訓練，最終模型準確率可再提升到 95.6%。

V. 實驗五：

- Epoch：15
- batch size：32
- CNN 激活函數：relu / 最後一層全連接層使用 sigmoid

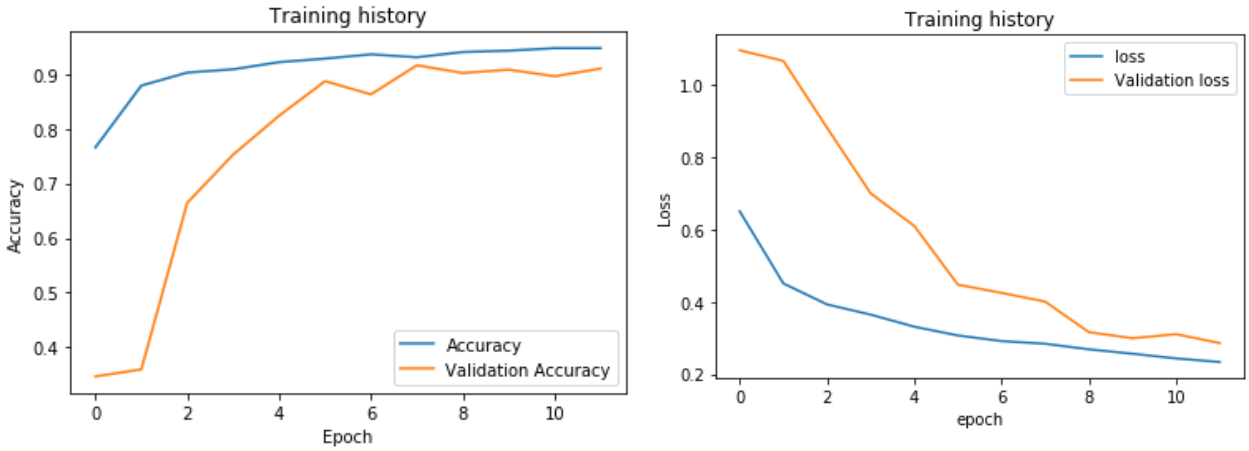
- Optimizer：Adagrad



共 125 個 batch 進行訓練，在第 13 個 epoch 時提早停止，最終模型準確率為 92.7%。

VI. 實驗六：

- Epoch：15
- batch size：64
- CNN 激活函數：relu / 最後一層全連接層使用 sigmoid
- Optimizer：Adagrad



共 63 個 batch 進行訓練，在第 12 個 epoch 時提早停止，最終模型準確率稍下降為 92.5%。

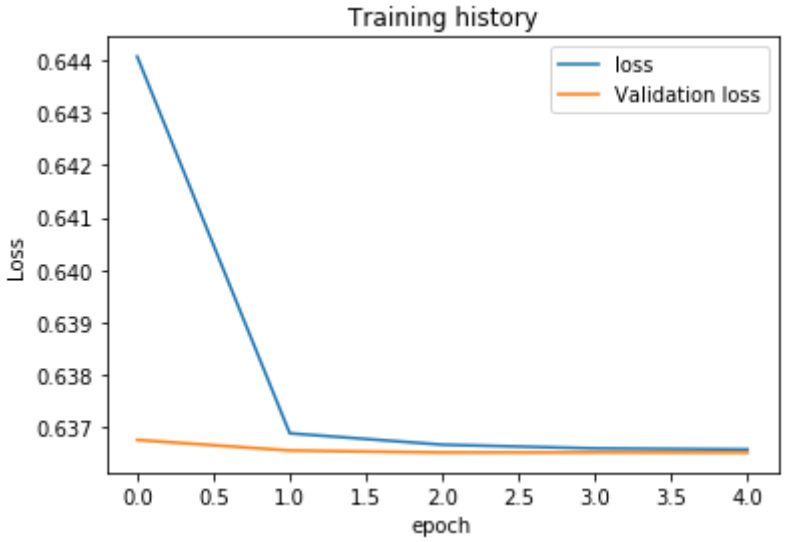
(2) LSTM 模型：

VII. 實驗七：

- Epoch：15
- batch size：32
- 激活函數：sigmoid

- Optimizer：adam

共 63 個 batch 進行訓練，在第 5 個 epoch 時提早停止，最終模型準確率為 66.7%



VIII. 實驗八：

- Epoch：15
- batch size：32
- 激活函數：softmax
- Optimizer：rmsprop

共 63 個 batch 進行訓練，在第 5 個 epoch 時提早停止，最終模型準確率為 33.3%

將上述所有實驗參數及結果整理為下表：

實驗編號	Network	Actual training epoch	batch size	Activate function	optimizer	Testing accuracy
1	CNN	9	32	Relu / sigmoid	adam	77.8%
2		15	32	Relu / softmax	adagrad	92.0%
3		11	32	Relu / softmax	adam	94.4%
4		15	64	Relu / softmax	adam	95.6%
5		13	32	Relu / sigmoid	adagrad	92.7%

6		12	64	Relu / sigmoid	adagrad	92.5%
7	LSTM	5	32	sigmoid	adam	66.7%
8		5	32	softmax	rmsprop	33.3%

五、 結論

本研究利用 CNN 及 LSTM 兩種深度學習模型進行語音辨識應用的實作，由多次調參的實驗結果可以發現，CNN 對於圖片資料的學習與辨識度較高，大多數皆有 9 成以上的準確率，經過優化後準確率更可達 95.6%。LSTM 雖有記憶特性，但在此例中相對表現較差，未來可考慮結合兩種神經網路，以期進一步提升模型準確率。

六、 參考資料

1. http://goofydayrecording.blogspot.com/2015/03/sample-ratebit-depth.html
2. http://mirlab.org/jang/books/audioSignalProcessing/audioIntro.asp?title=3-1%20Introduction%20to%20Audio%20Signals%20(%AD%B5%B0T%B0%F2%A5%BB%A4%B6%B2%D0)&language=chinese
3. https://ithelp.ithome.com.tw/articles/10192028
4. https://www.itread01.com/content/1550454495.html
5. https://kknews.cc/tech/ren98zn.html
6. https://medium.com/%E6%88%91%E5%B0%B1%E5%95%8F%E4%B8%80%E5%8F%A5-%E6%80%8E%E9%BA%BC%E5%AF%AB/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E8%87%AA%E5%8B%95%E8%AA%9E%E9%9F%B3%E8%BE%A8%E8%AD%98-3e2681b2e32c
7. https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47
8. https://medium.com/@chih.sheng.huang821/%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E8%B7%AF-convolutional-neural-network-cnn-cnn%E9%81%8B%E7%AE%97%E6%B5%81%E7%A8%8B-ecaec240a631
9. https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/467871/
10. https://dkopczyk.quantee.co.uk/speech-nn/