**臺北科技大學資訊工程系**

**107學年度實務專題計畫成果報告**

**長短期記憶神經網路於樂段生成之應用**

專題編號：107-CSIE-S013

專題計劃參與人員：104590024 蔡一玄

指導教授：尤信程 教授

執行期間：106年1學期至107年1學期

摘 要

　　近年來，人工智慧在科技領域中佔有舉足輕重的地位，未來也有很大的發展空間。人工智慧應用的領域十分廣泛，最廣為人知的幾個例子包括：物體及圖像識別、語音識別、天氣預測、市場動向預測、語言翻譯。而與人類創造力相關的事情，像是文學、繪畫、音樂創作，融入了人類的情感以及個人特質，加上成千上萬的變化性，以現今的技術來說，要讓機器學會這些事情是有相當難度的。

　　即便如此，筆者認為若是以現有的樂曲來進行模仿創作似乎可行，本專題即是利用人工智慧創作出仿巴洛克時期音樂風格的聖詠合唱曲。利用網路上所搜集到的MIDI檔，透過一些輔助工具擷取當中每個音符的資訊(音高、持續時間)，並轉換成電腦可辨識的數字表示法，用一個三維矩陣來儲存它們。接著使用深度學習中的LSTM神經網路模型，訓練後產生與輸入矩陣相似的輸出矩陣，再將輸出矩陣轉換為MIDI檔，得到的最終結果就是電腦仿作的樂曲。

目 錄

[摘 要 i](#_Toc525805689)

[目 錄 ii](#_Toc525805690)

[表目錄 v](#_Toc525805691)

[圖目錄 vi](#_Toc525805692)

[第一章 緒論 1](#_Toc525805693)

[1.1 研究背景及動機 1](#_Toc525805694)

[1.2 研究目標 1](#_Toc525805695)

[第二章 背景知識與使用技術 2](#_Toc525805696)

[2.1 基礎樂理 2](#_Toc525805697)

[2.1.1 音高與音程 2](#_Toc525805698)

[2.1.2 音符時值 2](#_Toc525805699)

[2.1.3 節拍與小節 3](#_Toc525805700)

[2.2音樂風格類型 3](#_Toc525805701)

[2.2.1 巴洛克時期 3](#_Toc525805702)

[2.2.2 古典樂派時期 4](#_Toc525805703)

[2.2.3 浪漫樂派時期 5](#_Toc525805704)

[2.2.4 現代樂派時期 5](#_Toc525805705)

[2.3 深度學習 6](#_Toc525805706)

[2.4 類神經網路 7](#_Toc525805707)

[2.5 長短期記憶神經網路簡介及架構說明 8](#_Toc525805708)

[2.5.1 簡介 8](#_Toc525805709)

[2.5.2 架構說明 10](#_Toc525805710)

[2.6 Numpy 簡介及基本使用方法 11](#_Toc525805711)

[2.6.1 簡介 11](#_Toc525805712)

[2.6.2 基本使用方法 12](#_Toc525805713)

[2.6.2.1 創建矩陣 12](#_Toc525805714)

[2.6.2.2 矩陣操作 14](#_Toc525805715)

[2.7 Keras簡介及基本使用方法 16](#_Toc525805716)

[2.7.1 簡介 16](#_Toc525805717)

[2.7.2 基本使用方法 16](#_Toc525805718)

[2.8 Music21簡介及基本使用方法 17](#_Toc525805719)

[2.8.1 簡介 17](#_Toc525805720)

[2.8.2 基本使用方法 18](#_Toc525805721)

[第三章 實作方法 20](#_Toc525805722)

[3.1 資料集選擇 20](#_Toc525805723)

[3.2 前置作業 20](#_Toc525805724)

[3.2.1 資料預處理 20](#_Toc525805725)

[3.2.2 資料擴增及資料分割 21](#_Toc525805726)

[3.2.3 資料編碼及表達方式 22](#_Toc525805727)

[3.3 利用LSTM神經網路預測音符 24](#_Toc525805728)

[3.4 音樂生成 26](#_Toc525805729)

[第四章 研究結果與分析 27](#_Toc525805730)

[4.1 兩種資料標記方法比較 27](#_Toc525805731)

[4.2 生成樂曲段落分析 28](#_Toc525805732)

[4.3 機器生成樂曲的瓶頸及困難之處 30](#_Toc525805733)

[4.3.1樂曲結構 30](#_Toc525805734)

[4.3.2 樂曲獨特性及品質的兼顧 32](#_Toc525805735)

[4.4 系統特色及價值 32](#_Toc525805736)

[第五章 結論與未來展望 34](#_Toc525805737)

[5.1 結論 34](#_Toc525805738)

[5.1.1 本系統待改進之處 34](#_Toc525805739)

[5.1.2 從本專題學習到的知識 35](#_Toc525805740)

[5.2 未來展望 35](#_Toc525805741)

[參考文獻 36](#_Toc525805742)

表目錄

[表2.1 音名及音高表 2](#_Toc525757421)

[表3.1 各編號及其代表意義 23](#_Toc525757422)

[表3.2 LSTM參數及其代表意義 27](#_Toc525757423)

圖目錄

[圖2.1 音符之間的關係圖 2](#_Toc525757257)

[圖2.2 拍號與小節範例(譜面選自Beethoven – piano sonata op.2, no.2, 1st mov.) 3](#_Toc525757258)

[圖2.3 巴洛克時期音樂譜例(選自Bach - keyboard partita BWV826 Sinfonia) 4](#_Toc525757259)

[圖2.4 漸強範例(選自Mozart – piano sonata k.310, mov.1) 4](#_Toc525757260)

[圖2.5 浪漫樂派樂曲範例(選自Chopin – Ballade No.4, op.52) 5](#_Toc525757261)

[圖2.6 無調音樂範例(選自Arnold Schoenberg – Piano concerto, op.42) 6](#_Toc525757262)

[圖2.7 人工智慧、機器學習、深度學習關係圖 6](#_Toc525757263)

[圖2.8 生物神經元與神經網路節點 7](#_Toc525757264)

[圖2.9 手寫數字及其特徵矩陣 8](#_Toc525757265)

[圖2.10 傳統神經網路 9](#_Toc525757266)

[圖2.11 遞歸神經網路 9](#_Toc525757267)

[圖2.12 LSTM模型結構(遺忘閘部分) 10](#_Toc525757268)

[圖2.13 LSTM模型結構(輸入閘部分) 10](#_Toc525757269)

[圖2.14 LSTM模型(Cell State更新部分) 11](#_Toc525757270)

[圖2.15 LSTM模型(輸出閘部分) 11](#_Toc525757271)

[圖2.16 創建一維陣列 12](#_Toc525757272)

[圖2.17 創建二維陣列 12](#_Toc525757273)

[圖2.18 創建充滿0的2x5陣列 12](#_Toc525757274)

[圖2.19 創建充滿1的2x5陣列 13](#_Toc525757275)

[圖2.20 創建固定範圍或間隔陣列 13](#_Toc525757276)

[圖2.20 創建一個5x5的單位矩陣 13](#_Toc525757277)

[圖2.21 創建一個2x3的隨機陣列 14](#_Toc525757278)

[圖2.22 將一個2x3陣列重排 14](#_Toc525757279)

[圖2.23 壓縮矩陣範例 14](#_Toc525757280)

[圖2.24 轉置矩陣範例 15](#_Toc525757281)

[圖2.25 串接兩矩陣範例 15](#_Toc525757282)

[圖2.26 分割陣列範例 16](#_Toc525757283)

[圖2.27 Keras使用範例－part1 17](#_Toc525757284)

[圖2.28 Keras 使用範例－part2 17](#_Toc525757285)

[圖2.29 Keras 使用範例－part3 17](#_Toc525757286)

[圖2.30 模型預測結果(上方陣列為輸入資料，下方為模型預測的結果) 18](#_Toc525757287)

[圖2.30 載入音檔並顯示樂譜 18](#_Toc525757288)

[圖2.31 讀取各聲部名稱 19](#_Toc525757289)

[圖2.32 讀取各聲部總長度 19](#_Toc525757290)

[圖2.33 讀取第二聲部各音符的名稱、midi音符編號、持續時間 19](#_Toc525757291)

[圖2.34 取得樂曲調性及轉調範例 20](#_Toc525757292)

[圖3.1 檢查聲部與過濾音檔 22](#_Toc525757293)

[圖3.2 聲部長度檢查及調整長度 22](#_Toc525757294)

[圖3.3 將輸入樂曲上下各轉調半音 22](#_Toc525757295)

[圖3.4 分割資料函式 23](#_Toc525757296)

[圖3.5 擷取樂曲音符 24](#_Toc525757297)

[圖3.6 處理編碼及解碼的函式 24](#_Toc525757298)

[圖3.7 LSTM 與輸入資料關係示意圖 25](#_Toc525757299)

[圖3.8 數字轉換為one-hot表示法示意圖 25](#_Toc525757300)

[圖3.9 模型架構 26](#_Toc525757301)

[圖3.10 初始化模型程式碼 26](#_Toc525757302)

[圖3.11 輸入資料及目標資料格式示意圖 27](#_Toc525757303)

[圖4.1 神經網路訓練過程的訊息(方法一) 29](#_Toc525757304)

[圖4.2 神經網路訓練過程的訊息(方法二) 30](#_Toc525757305)

[圖4.3 方法一的樣本樂譜範例 31](#_Toc525757306)

[圖4.4 輸入音樂樣本範例 31](#_Toc525757307)

[圖4.5 輸入音樂與機器生成音樂的主旋律片段 31](#_Toc525757308)

[圖4.6 方法二的樣本樂譜範例 32](#_Toc525757309)

[圖4.7 輸入音樂譜例一 33](#_Toc525757310)

[圖4.8 輸入音樂譜例二 33](#_Toc525757311)

[圖4.9 機器生成樂曲樣本 34](#_Toc525757312)

第一章 緒論

1.1 研究背景及動機

　　筆者剛接觸資工領域之時，便對人工智慧領域產生好奇，但是那時候認為人工智慧應該是高階工程師或是資工方面專家研究的領域，自己還是個剛起步的初學者，應當先打好基礎並學習一些前置的知識。在大學一二年級之間，筆者修習了本系開設的程式及數學的基本課程，從中學習到許多必要的知識，像是線性代數、物件導向程式設計、資料結構。在完成這些前置課程之際，筆者認為自己可以嘗試踏入人工智慧領域，於是就想利用大學畢業專題的機會，製作一個與人工智慧相關的專題。

　　另外，由於筆者會利用課餘時間練習鋼琴以及研究音樂，便想到可以將自己的興趣與專業領域的技術融合，做出一個非常獨特的專題，並藉由這個專題學習一些人工智慧方面的知識，以及探究人工智慧的發展可能性。

1.2 研究目標

　　本專題最終的目標為讓電腦創作出一段與特定樂曲風格類似的曲子，由於世界上的樂曲非常多，加上不同時期、風格的樂曲難易度與複雜度皆不盡相同，因此首要的目標就是找到一種適合電腦模仿的樂曲類型。在確定好樂曲類型後，接著需要決定的是實作方法。機器學習中有各式各樣的技術以及演算法，每一種適合應用的領域也不同，因此第二個目標必須要找到一個易於電腦仿作音樂的方法及系統框架。成功讓電腦製造出樂曲後，還需要進行樂曲好壞以及重複性的評估，盡可能讓電腦創作的音樂具有獨特性以及和諧的旋律。

第二章 背景知識與使用技術

2.1 基礎樂理

2.1.1 音高與音程

　　樂曲中，每個音符都有各自的音高，每個音高會對應到一個固定的頻率，而音高的種類一共十二個[1]，如表2.1(以C Major為例)：

表2.1 音名及音高表

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 音名 | C | C♯/D♭ | D | D♯/E♭ | E | F | F♯/G♭ | G | G♯/A♭ | A | A♯/Bb | B |
| 頻率(Hz) | 262 | 277 | 294 | 311 | 330 | 349 | 370 | 392 | 415 | 440 | 466 | 494 |

　　音符之間的音高差距為音程，音程的單位為度。例如，C與D為二度，C與E為三度，C到下個較高音的C為八度。

2.1.2 音符時值

　　音符時值用來表示音符持續時間比例的關係，實際的速度是由速度標記(Tempo)所決定，基本的時值詳見圖2.1：

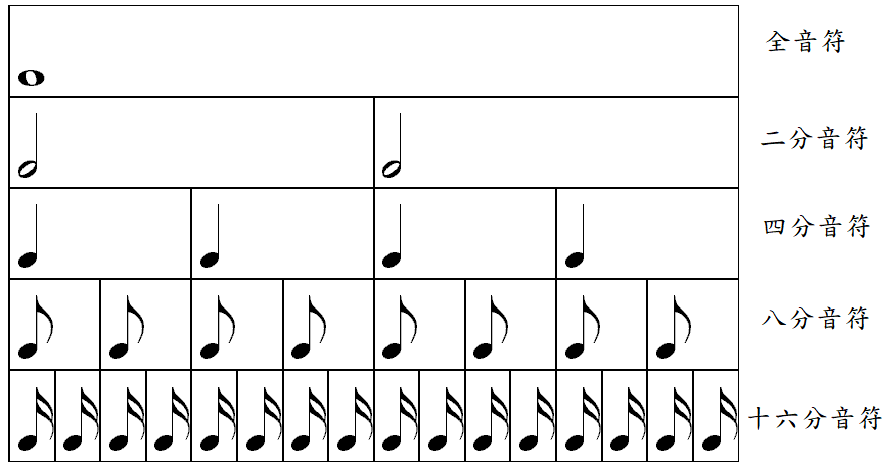


圖2.1 音符之間的關係圖

2.1.3 節拍與小節

　　通常在樂譜的開頭會標示拍號，以分數的形式呈現[1]，詳細的說明如圖2.2。

　　小節的功用為使樂曲的旋律結構更明確，我們會在小節與小節中間以小節線來分隔(如圖2.2)，每個小節的拍數皆相同。



圖2.2 拍號與小節範例(譜面選自Beethoven – piano sonata op.2, no.2, 1st mov.)

2.2音樂風格類型

　　為了找出較適合電腦模仿的樂曲風格，必須先了解各時期的樂曲結構及特色，以年代做為區分大致可分為巴洛克時期、古典樂派時期、浪漫樂派時期、現代樂派時期，以下分別做詳細說明。

2.2.1 巴洛克時期

　　這個時期的樂曲結構較明確，作曲的規則較嚴謹，在力度方面的層次變化不大，節奏方面有著較固定的拍子。圖2.3就是一個典型的例子，動機的長短都是以一個小節為單位，拍子的部分也非常規律，聲部之間的對位也十分明顯。

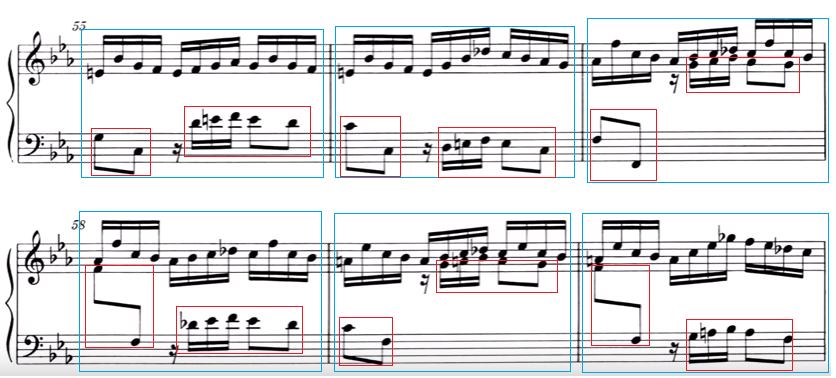


圖2.3 巴洛克時期音樂譜例(選自Bach - keyboard partita BWV826 Sinfonia)

2.2.2 古典樂派時期

　　巴洛克時期因為大多曲子為複音音樂，必須保持聲部之間的平衡，故不能有太大的強弱起伏，但從古典樂派時期開始，樂句便有了漸進的強弱表現[2]。另一個特色是，奏鳴曲式在古典時期被廣泛的採用，而奏鳴曲是一種具有邏輯性以及明確架構的曲式，主題及段落的安排富有戲劇性，就好像在寫一篇作文一樣，會有起、承、轉、合四個段落，而奏鳴曲則是分為呈示部、發展部、再現部[2]。



圖2.4 漸強範例(選自Mozart – piano sonata k.310, mov.1)

2.2.3 浪漫樂派時期

　　巴洛克及古典時期的作曲家都是為了皇室或是宗教作曲，樂曲當中便比較沒辦法融入個人的情感，也因為一些作曲上的規範而讓樂曲有一定的框架或是風格存在。但是到了浪漫樂派，作曲家們可以將自己的想法及感受融入樂曲，或是以某人、事或物做為主題創作一首曲子。而在樂曲的形式上雖然承襲了古典時期，但在樂曲內容方面就比較不受拘束及限制[2]。



圖2.5 浪漫樂派樂曲範例(選自Chopin – Ballade No.4, op.52)

2.2.4 現代樂派時期

　　現代樂派早期，大多作曲家還是依循浪漫時期的作曲風格，只是後來人們開始覺得音樂的變化應該要再更豐富一點，曲式以及和聲的運用要更大膽開放[2]，因此，不和諧的和弦配置以及打破傳統調性的音樂(無調音樂，見圖2.6)便誕生了。而許多新的音樂形式像是電子音樂、嘻哈舞曲、鄉村音樂、爵士音樂也逐漸興起，讓整個音樂的世界變得更多樣化。

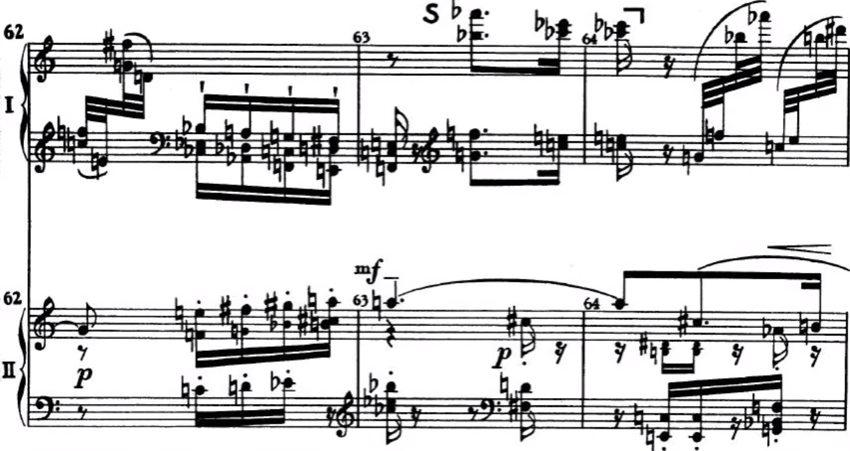


圖2.6 無調音樂範例(選自Arnold Schoenberg – Piano concerto, op.42)

2.3 深度學習

　　深度學習、機器學習以及人工智慧這三個名詞是大家很容易混淆的，為了知道它們之間的關係，我們可以簡單用一張圖來表示(如圖2.7)：

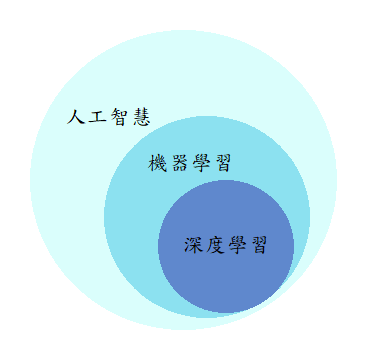


圖2.7 人工智慧、機器學習、深度學習關係圖

　　簡單來說，人工智慧廣泛涵蓋了所有擁有人類智慧的機器，而機器學習只是其中一個實踐人工智慧的方法，基本上是運用各種演算法來達到我們的需求。深度學習是機器學習中的一個分支，運用了類神經網路的技術來讓訓練效果更好，預測的結果更準確。

　　由於傳統的機器學習方法在許多的分類及預測工作上已經到了臨界值，準確度很難再上升，因此後來發展出深度學習的技術來改善這個問題，這也是為甚麼現今人們極力於發明新的神經網路，讓準確度盡可能接近百分之百。

2.4 類神經網路

　　深度學習所使用的神經網路基本的架構是模仿生物大腦的神經系統。在深度學習的神經網路中，主要可以分為三層：輸入層(input layer)、隱藏層(hidden layer)、輸出層(output layer)，每一層都是由許多節點組成的，而這些節點就像是大腦神經系統中的神經元，輸入的資料就如同神經元接收訊息再傳給其他神經元。從圖2.8中我們可以發現大腦神經系統中的神經元與神經網路的節點之間的相似之處。

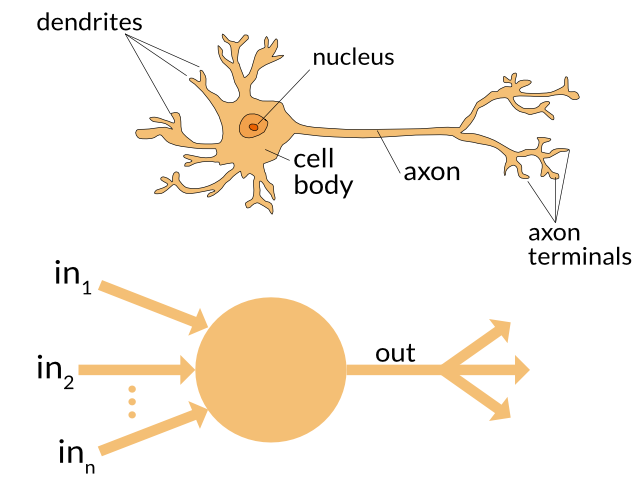


圖2.8 生物神經元與神經網路節點

　　以手寫數字辨識來說，人類藉由眼睛接收到數字的外觀及特徵，經由視神經傳達至大腦，大腦接收到訊息後便可以判斷所見的數字類別。對於神經網路而言，我們會將大量的手寫數字圖片以矩陣的表示方式，矩陣中的數字代表圖片中顏色的深淺度(範例如圖2.9)為了讓機器能夠藉由特徵來辨認數字，我們必須將每張圖片進行標記，也就是讓圖片有個標準答案。在訓練神經網路的過程中，機器會將圖片的特徵向量送進輸入層，經過一連串的運算後，輸出層會得到預測的結果。然而，只有一輪的訓練是不可能預測準確的，因此，我們會使用代價函數(Cost Function)來「對答案」，也就是評估預測的結果與真實之間的差距。得到這個差距值後，利用反向傳播(Backward-Propagation)演算法將預測的結果傳遞回去以調整神經元的權重，讓下一輪訓練預測的結果更準確。

　　也因為神經網路需要經過很多輪的訓練，每次訓練都像是人類學習過程中會不斷進行修正，才會把這樣的方法歸類為「深度學習」。

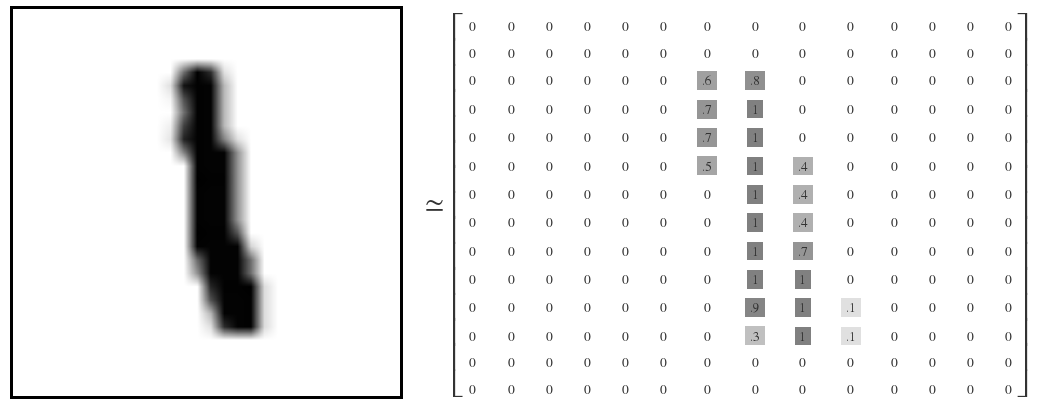


圖2.9 手寫數字及其特徵矩陣

2.5 長短期記憶神經網路簡介及架構說明

2.5.1 簡介

　　傳統的神經網路將每一層的輸出接到下一層的輸入，資料僅能單方向由輸入層傳遞至輸出層(結構如圖2.10)，因此並無法將資料中的訊息保留於某一層當中，輸出的結果自然就決定於輸入的結果。後來資料科學家發明了遞歸神經網路(Recurrent Neural Networks)，這類的神經網路具有記憶性，資料在傳遞的過程中，重要的訊息會記錄於記憶單元(Memory Cell)，因此輸出便不再只由輸入決定，每一層的狀態也會影響到輸出的結果(結構如圖2.11)。有了這個記憶的特性，讓遞歸神經網路更適合處理連續且有關聯性的資料。就像有些事情我們若僅從當下的情況做判斷，往往無法得到較正確的結果，但是當我們有了歷史資料，便能藉由這些資料較精準預測出結果。

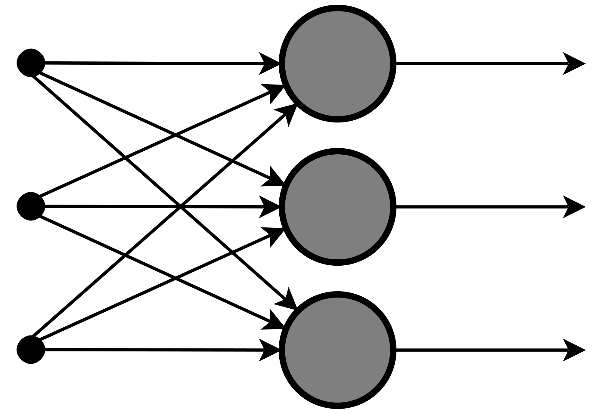


圖2.10 傳統神經網路

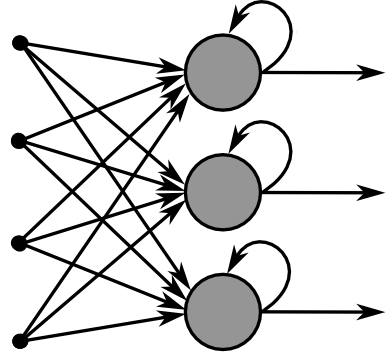


圖2.11 遞歸神經網路

　　然而，一般的遞歸神經網路容易衍生梯度消失(Gradient Vanishing)的問題，這個問題會導致較久以前的資料變得不重要，也就是神經網路無法有較長期的記憶，而長短期記憶神經網路(Long Short-Term Memory Neural Network, LSTM)就是一個解決這個問題的方法。

2.5.2 架構說明

　長短期記憶神經網路主要由輸入閘(Input Gate)、遺忘閘(Forget Gate)、輸出閘(Output Gate)組成，以下分別說明這三種閘的功用以及原理：

1. 遺忘閘：功能為決定cell state的遺忘程度。首先，輸入的資料(xt)與前一時刻的hidden state(ht-1)會經過一個sigmoid 函數，並輸出一個介於0~1之間的數字(ft)，而這個數字就是決定前一個cell state(Ct-1)的被遺忘程度，若為0表示完全捨棄，1則表示完全保留。遺忘閘的計算公式及結構如圖2.12所示。

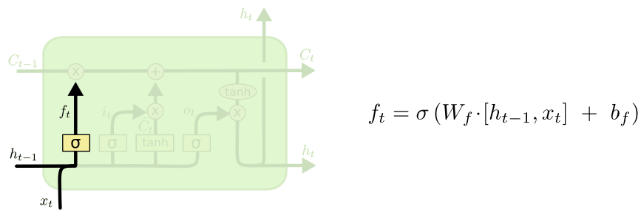


圖2.12 LSTM模型結構(遺忘閘部分)

1. 輸入閘：負責決定哪些資料要儲存進cell state中。首先將輸入資料(xt)與前一時刻狀態(ht-1)分別透過sigmoid及tanh函數運算，分別會得到it及C͂t 這兩個值，其中C͂t為候選的更新值。輸入閘及候選更新值的計算方式及結構如圖2.13所示。

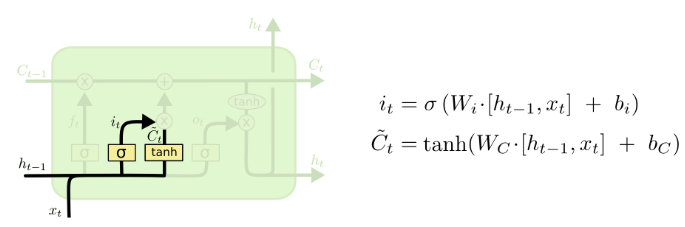


圖2.13 LSTM模型結構(輸入閘部分)

　　接著我們將舊的cell state值進行更新，首先把遺忘閘產生的結果(ft)與前一個cell state(Ct-1)相乘，再加上輸入閘的結果(it)與候選更新值(C͂t)相乘後的值，即可得到新的cell state(Ct)。(如圖2.14)

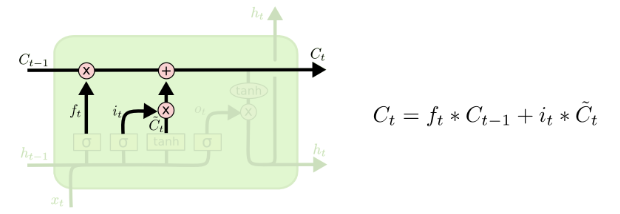


圖2.14 LSTM模型(Cell State更新部分)

1. 輸出閘：這個閘輸出的值取決於當下儲存於cell state(Ct)中的值。首先，我們將xt及ht-1的值做為過濾器，這個過濾器可以過濾掉我們不想要的值，另一方面，Ct會經由tanh函數讓其值壓縮至-1~1之間。接著我們將過濾器產生出來的值與Ct經壓縮過後的值相乘， 即可得到最終的輸出結果，而這個也將此刻的hidden state做更新。

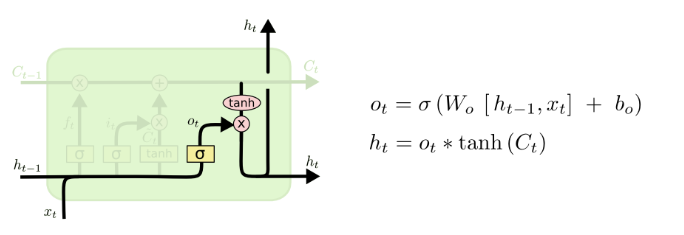


圖2.15 LSTM模型(輸出閘部分)

2.6 Numpy 簡介及基本使用方法

2.6.1 簡介

　　Numpy 是一個基於Python語言撰寫的函式庫，在資料科學領域被廣泛的使用，其主要功能為提供使用者進行高維的矩陣運算，並包含了大量的數學運算函式，而執行效率方面也較Python本身提供的陣列運算還高很多。除此之外，Numpy亦可結合其他的Python函式庫使用，像是與Matlab功能相似的Scipy，或是擁有強大繪圖功能的Matplotlib。

2.6.2 基本使用方法

　　以下筆者分成兩個部分介紹一些Numpy的常見用法，包含創建矩陣以及矩陣操作。

2.6.2.1 創建矩陣

* Numpy.array: 創建一個Numpy陣列，傳入的資料型態為一個Python中的標準陣列。

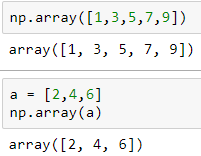


圖2.16 創建一維陣列

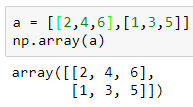


圖2.17 創建二維陣列

* Numpy.zeros: 創建一個全為0的Numpy陣列。

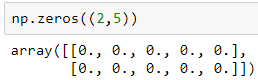


圖2.18 創建充滿0的2x5陣列

* Numpy.ones: 創建一個全為1的 Numpy陣列。

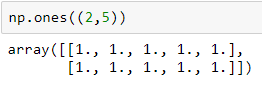


圖2.19 創建充滿1的2x5陣列

* Numpy.arange: 創建一個內容為固定範圍或間隔的陣列。

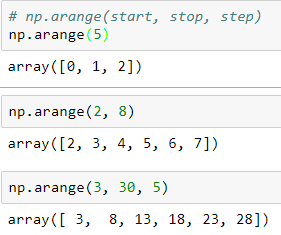


圖2.20 創建固定範圍或間隔陣列

* Numpy.identity: 創建一個單位矩陣。

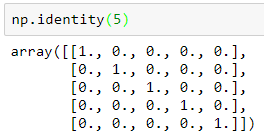


圖2.20 創建一個5x5的單位矩陣

* Numpy.random.rand: 創建一個內容隨機且介於[0,1)區間的陣列。

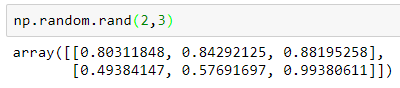


圖2.21 創建一個2x3的隨機陣列

2.6.2.2 矩陣操作

* Numpy.reshape: 在不改變陣列內容的情況下，重新調整陣列的行列數。

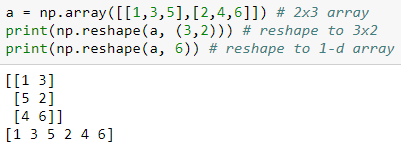


圖2.22 將一個2x3陣列重排

* Numpy.squeeze: 去除陣列中shape為1的元素。

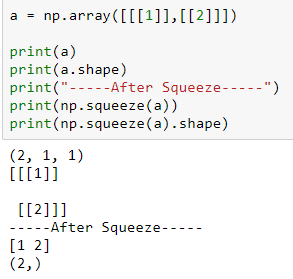


圖2.23 壓縮矩陣範例

* Numpy.transpose: 轉置一個矩陣。

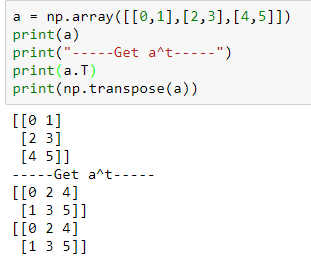


圖2.24 轉置矩陣範例

* Numpy.concatenate: 串接兩個陣列。

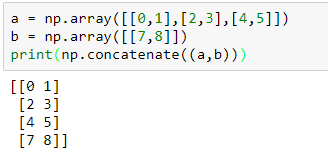


圖2.25 串接兩矩陣範例

* Numpy.split: 分割一個陣列，可設定各子陣列的大小，亦可設定分割點。

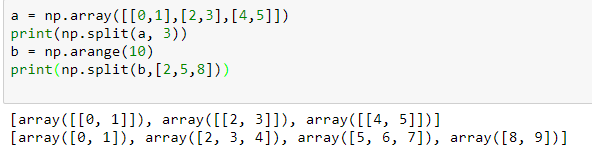


圖2.26 分割陣列範例

2.7 Keras簡介及基本使用方法

2.7.1 簡介

　　Keras 是一個基於Python撰寫而成的高階神經網路API，由於可以在幾個機器學習API上運行(例如：TensorFlow、CNTK、Theano)，因此對於一個剛學習神經網路的初學者來說，Keras是非常好上手的。

　　而Keras主要處理神經網路模型的創建、訓練及預測，其餘較底層的矩陣運算以及神經網路內部複雜運算會交給底部的機器學習套件，因此，程式碼的架構能簡化許多，同時也增加了可讀性。

2.7.2 基本使用方法

　　Keras的核心架構為一個模型(model)，而這個模型可以由多個層(layers)組成，最基本的模型類別是一個被稱作Sequential的模型。以下介紹使用Sequential 模型來將現有資料進行分類的範例：

　　首先隨機產生一萬筆資料，這些資料以一個特徵矩陣(x\_data)表示，並用另一個陣列進行標記(y\_labels)，作為分類的依據，而為了使機器更容易分類，我們將原本的標記陣列以one-hot的形式表示。(如圖2.27，在one-hot陣列中，每個種類皆代表一個元素，被分配到的類別標記為”1”，其餘的則為”0”)

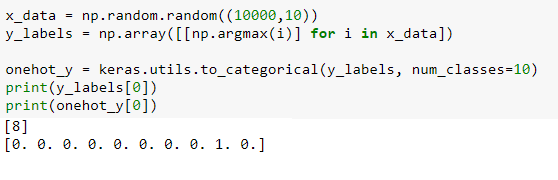


圖2.27 Keras使用範例－part1

　　接著，我們創建一個Sequential模型，其輸入層有10個節點，隱藏層有256個節點，輸出層有10個節點，其中，輸出層的節點數目一定要等於類別的數目。而激勵函數(Activation Function)使用Softmax的原因是，Softmax比較適合進行多類別的分類，從Softmax產生的矩陣中，我們可以得知各類別的機率分布。

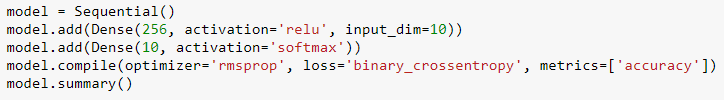


圖2.28 Keras 使用範例－part2

　　最後，我們將資料矩陣與標記矩陣送往模型進行訓練並預測，預測的結果顯示於圖3.30。

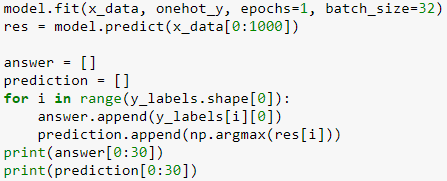


圖2.29 Keras 使用範例－part3



圖2.30 模型預測結果(上方陣列為輸入資料，下方為模型預測的結果)

2.8 Music21簡介及基本使用方法

2.8.1 簡介

Music21 是一個使用Python 語言撰寫的電腦輔助音樂學分析工具包，讓使用者透過簡易的程式碼解決音樂學方面的問題，或是針對音檔進行處理、解析，甚至可以使用程式碼來作曲或編曲。

2.8.2 基本使用方法

* 載入音檔：將欲載入的音檔路徑設定於converter.parse()中的參數。



圖2.30 載入音檔並顯示樂譜

* 讀取音檔聲部資訊：聲部的資訊由Part()管理，我們可以藉由它得知各聲部的名稱、時間長度、所有音符．．．等訊息。

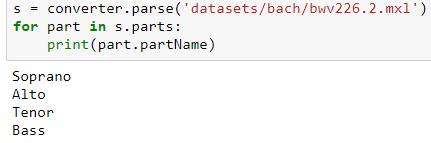


圖2.31 讀取各聲部名稱

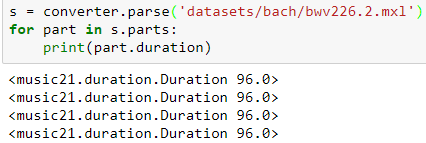


圖2.32 讀取各聲部總長度

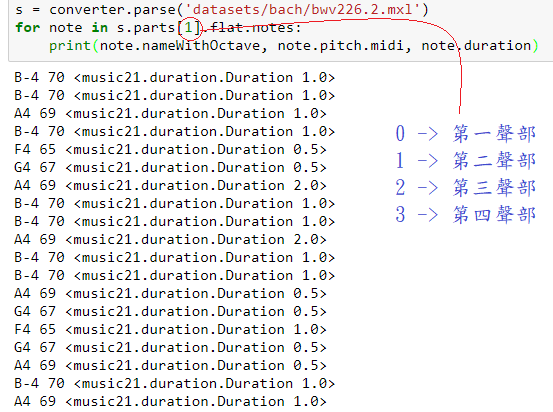


圖2.33 讀取第二聲部各音符的名稱、midi音符編號、持續時間

* 取得輸入樂曲的調性及將樂曲轉調方法：將一樂曲音檔轉為stream的資料型態後，使用analyze(‘key’) 即可得到該樂曲的調性。轉調部分則使用transpose(steps)，其中，steps表示欲轉調的半音數目(例如：step = -3表示向下轉3個半音，step = 2 表示向上轉2個半音)。

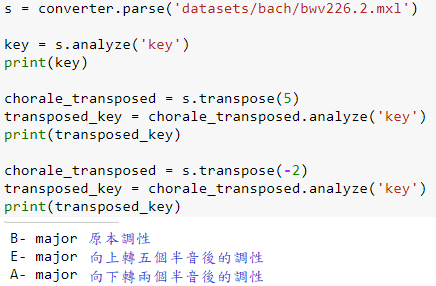


圖2.34 取得樂曲調性及轉調範例

第三章 實作方法

3.1 資料集選擇

　　為了讓電腦製造出與某種特定風格樂曲相仿的樂段，首先面臨的問題就是要選擇何種風格的樂曲作為資料集，我們必須讓資料集裡的樂曲風格盡量相近。另外，還需要考量樂曲的複雜程度，越複雜的曲子會有更多的細節，模仿的難度相對較高，因此，筆者決定先採用結構較簡易的樂曲。

　　在第二章第二節所介紹的各時期樂曲風格中，以樂曲的結構來說，巴洛克時期及古典樂派時期相對簡單，不過古典時期在力度的變化上比巴洛克時期還要多，因此筆者最後決定採用巴洛克音樂的資料集。巴洛克時期的音樂家當中，以約翰．賽巴斯蒂安．巴哈 (J. S. Bach) 最為著名，其所作的樂曲數量甚多。而在現有的巴哈樂曲資料集以聖詠合唱曲最為齊全，一共約400首，加上網路上也有其他人使用這個資料集讓電腦進行重配和聲，因此筆者認為可以先採用此資料集。

3.2 前置作業

在準備好資料集之後，我們需要將這些音檔轉換成電腦可以讀取的型態，原本這件工作是非常麻煩的，但有了music21這個套件的幫助，僅需幾行程式碼即可完成，以下筆者會詳細說明如何進行輸入資料處理。

3.2.1 資料預處理

首先，為了避免資料不一致的情形發生，我們訂定了兩個基本的規則，第一，樂曲必須為四聲部，第二，樂曲中每個聲部的長度必須一致。若為非四聲部的樂曲，便不採用該樂曲；若樂曲中每個聲部長度不一致，則會使用休止符將每個聲部的長度補至與時間最長的聲部相同。(程式碼實作如圖3.1及3.2所示)

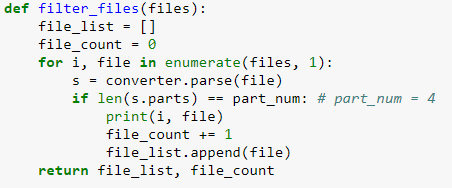


圖3.1 檢查聲部與過濾音檔

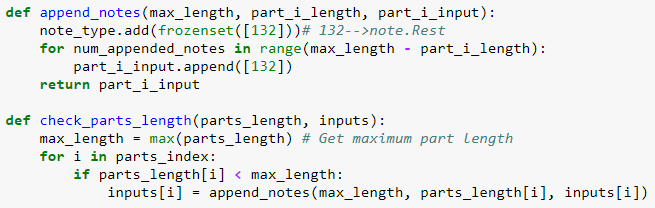


圖3.2 聲部長度檢查及調整長度

3.2.2 資料擴增及資料分割

　　由於資料集的樂曲數量僅有約400首，理想上越多資料會得到比較好的結果，而以音樂來說，使用轉調來進行資料擴增是最常見的方式。

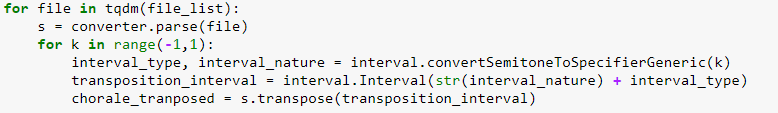


圖3.3 將輸入樂曲上下各轉調半音

　　資料分割的部分，我們將資料集以8:2的比例拆成訓練資料及測試資料，訓練資料是我們在訓練神經網路作為輸入的資料，而測試資料是當我們要預測音符及產生結果時用的資料。

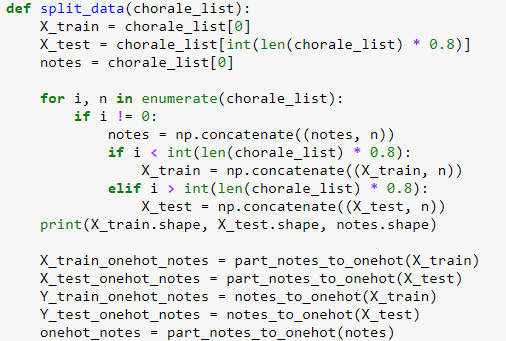


圖3.4 分割資料函式

3.2.3 資料編碼及表達方式

　　一個好的資料表示方法可以讓神經網路的訓練效果及得到的成果較好，本系統的資料表示法參考Gaetan Hadjeres的DeepBach[10]，並加以簡化。首先我們將樂曲的四個聲部分別擷取每個音符的MIDI編號，並存入對應的四個陣列中，而這些音符有可能以單音或是和弦的方式呈現，因此需要用一個set來儲存。另外，由於樂曲中除了音符以外還有其他元素，像是休止符或是持續音，因此會需要額外的符號來表示它們。(資料表達方式如表3.1，程式碼如圖3.5)

表3.1 各編號及其代表意義

|  |  |
| --- | --- |
| 0~127 | 音高，對應於MIDI編號 |
| 128~131 | 音符的持續符號（例如：73 128，代表73這個音長度為兩拍），而128表示第一聲部，129表示第二聲部，以此類推 |
| 132 | 休止符 |
| 133、134 | 樂曲的起始符號及終止符號 |



圖3.5 擷取樂曲音符

　　為了讓資料的表達方式更為簡化，我們將各音符及和弦的種類進行編碼，每種音符或和弦會對應到一個編號。而當訓練完畢時，我們也必須將這些編號解碼還原成音符及和弦，以便輸出新的音檔，因此，我們會需要一個函式處理音符與和弦的編碼及解碼(如圖3.6)。

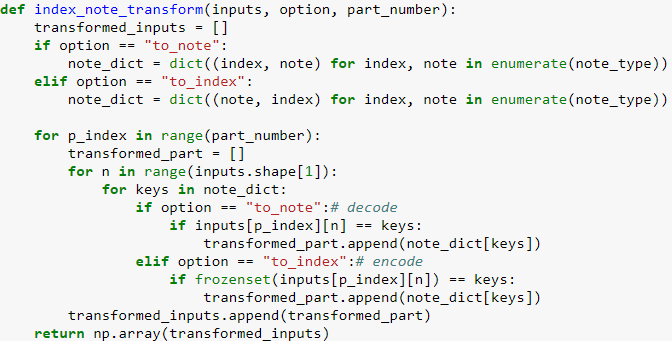


圖3.6 處理編碼及解碼的函式

　　由於LSTM模型輸入為同一時間各聲部的資訊(圖3.7為示意圖)，我們必須利用陣列索引來取得某個時間點四個聲部的內容，為了方便操作，我們將原本以聲部當作第一維度的矩陣轉置(Transpose)為以時間當作第一維度的矩陣，如此一來，第一維度的索引便是一個單位時間長。

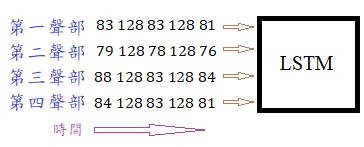


圖3.7 LSTM 與輸入資料關係示意圖

　　最後，為了讓神經網路在運算及預測時有較佳的結果，我們將陣列中的數字轉換為於2.7.2小節提及的one-hot表示法，轉換過後的陣列即為LSTM神經網路的輸入資料。

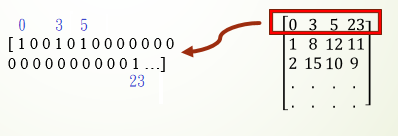


圖3.8 數字轉換為one-hot表示法示意圖

3.3 利用LSTM神經網路預測音符

　　在設定好輸入資料的格式後，接下來的工作就是將這些資料分批送入LSTM模型進行訓練。在訓練之前，我們必須先定義好神經網路的訓練模型，本系統採用最基本的兩個LSTM神經網路作為訓練模型，相對於一個LSTM，兩個LSTM更能夠處理較複雜的輸入資料格式，也能有較好的預測結果。在兩個LSTM神經網路之後我們會接上一層Dropout Layer[11]，透過刪除節點的方式讓神經網路能夠使用不同節點來進行學習，以避免在訓練過程中過度依賴某些節點，而導致產生過擬合(Overfitting)的問題。輸出層的部分使用的是全連接層(Fully Connected Layer) ，會基於前面神經網路所產生的訊號來預測音符。而輸出層的激勵函數採用的是在2.7.2小節介紹的Softmax，可以得知音符的機率分布。(模型的架構如圖3.9所示，初始化模型的程式碼如圖3.10所示)

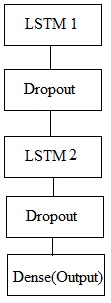


圖3.9 模型架構

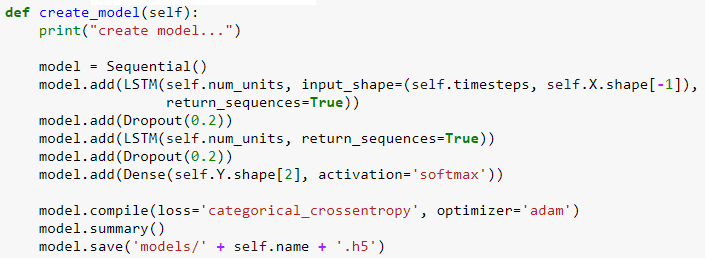


圖3.10 初始化模型程式碼

　　在訓練模型的時候，我們需要提供輸入資料的對應標記，而目標資料(拿來當標記的資料)的格式與輸入資料類似，唯一不同之處是，輸入資料經過轉置後所有聲部的訊息已被混合在一個陣列當中，但在目標資料中，我們為了讓機器分辨某時間點的音符是屬於哪個聲部的，因此將四個聲部的音符拆成四個陣列表示。(如圖3.11)

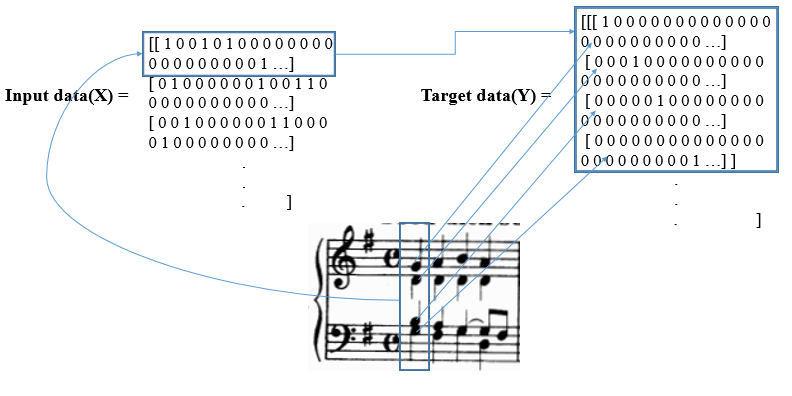


圖3.11 輸入資料及目標資料格式示意圖

　　神經網路創建及訓練的部分，Keras提供了很完善的功能，我們僅需了解其提供的參數即可，表3.2列出這些參數及其代表的意義。

表3.2 LSTM參數及其代表意義

|  |  |
| --- | --- |
| x | 輸入資料，型態為Numpy Array |
| y | 目標資料，型態為Numpy Array |
| num\_units | 神經網路層的節點個數 |
| epochs | 訓練全部資料的次數 |
| batch\_size | 每次訓練從資料集取的樣本批數 |
| timesteps | 訓練過程中單位時間的資料量 |
| validation\_data | 用來評估模型準確度的資料 |

3.4 音樂生成

　　訓練完模型之後，我們便可以利用Keras提供的模型預測函式來產生音符的機率矩陣，而透過這個機率矩陣我們可以得知每個時間點的音符機率分布，並取得陣列中機率最高的索引，接著利用這個索引即可解碼出對應的音符或和弦，將它們串接起來後再透過music21所提供的函式轉換為midi檔案，而這個檔案就是電腦生成的音樂。

第四章 研究結果與分析

4.1 兩種資料標記方法比較

　　由於標記的方法會直接影響到輸出音樂的結果與品質，因此必須找出一個比較合適的標記方法。

　　第一種方法是僅將聲部訊息被混合的輸入資料做聲部分類，不做任何時間上的位移，這樣的結果雖然能夠讓神經網路的訓練準確度達到95%以上(由圖4.1我們可以發現，大概訓練了五個epochs後模型準確度就高達90%以上)，但卻導致生成的音樂與輸入音樂相似度非常高。與指導教授請教後，發現這個方法用在音訊或是影像辨識會比較合適，但在電腦作曲方面，我們理想上會希望輸出的音樂僅保有輸入音樂的風格，而旋律及伴奏的部分需要有一些變化。

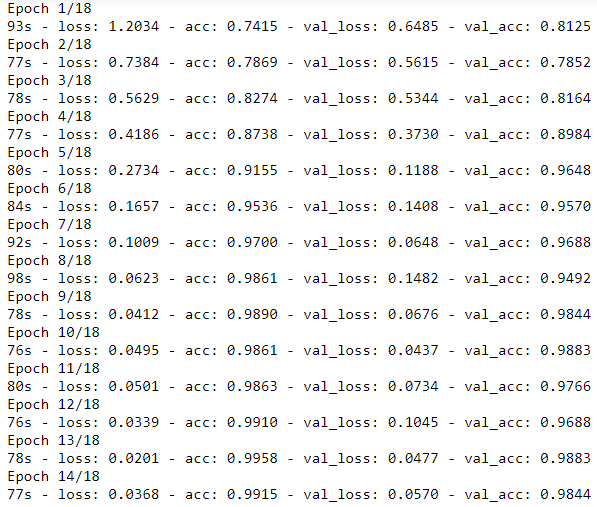


圖4.1 神經網路訓練過程的訊息(方法一)

　　第二種方法是將目標資料做一個小節的位移，也就是當輸入資料為樂曲中的第一小節時，目標資料即為樂曲的第二小節。而以一個小節做位移的原因是，通常一個小樂句或動機的長度差不多是一個小節，這麼做較可以避免產生的音樂中有一些不合常理的節奏。在模型訓練準確度的部分，我們可以從圖4.2發現與第一種方法相比低了一些，由此可知，機器生成的樂曲並不會與輸入音樂有太多的相似之處。雖然這樣的方法可以避免產生與輸入音樂極為類似的旋律，但在整體的樂曲品質方面還是有待改進。

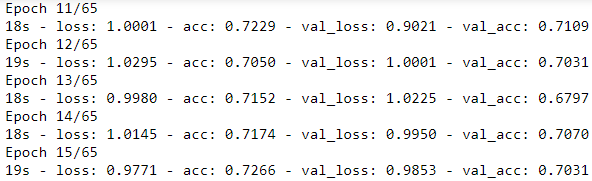


圖4.2 神經網路訓練過程的訊息(方法二)

4.2 生成樂曲段落分析

　　在上一個章節中，筆者簡單比較了兩種標記法在結果上的差異，而這一章節筆者會針對兩種方法所生成的樂曲進行分析。

　　使用第一種方法生成的樣本樂譜如圖4.3所示，從圖中我們可以看到基本的樣貌以及節拍都與輸入音樂(譜例如圖4.4)相似，尤其是主旋律部分(如圖4.5)我們可以看到機器很成功的模仿輸入音樂。



圖4.3 方法一的樣本樂譜範例



圖4.4 輸入音樂樣本範例

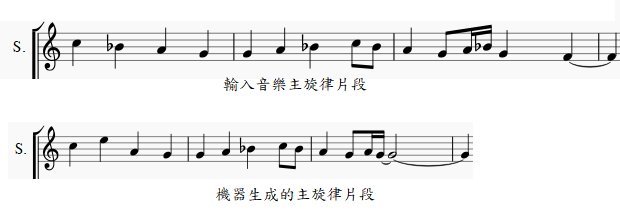


圖4.5 輸入音樂與機器生成音樂的主旋律片段

　　另一部分，使用第二種方法產生的音樂樣本中，雖然有一些片段存在不和諧的和聲以及較不規則的旋律配置，但還是能看到一些具有與輸入音樂風格類似又不失獨特性的段落。從圖4.6我們可以發現，樂曲當中會不時出現一些較特殊的和聲(縱向方框)，這些和聲可能不適合出現在這個段落，或是違反了一些樂理規則，因此聽起來會有違和感。同樣的，樂曲的片段中也有一些較不和諧的旋律，筆者在圖4.6中將這些旋律使用橫向方框標記。



圖4.6 方法二的樣本樂譜範例

4.3 機器生成樂曲的瓶頸及困難之處

4.3.1樂曲結構

　　在輸入音樂當中，我們可以看到一些較有規律性的樂段，讓聆聽者對這個段落會有較深刻的印象，由圖4.7及4.8可以發現，這種規律性的樂段長度大約為四個小節。



圖4.7 輸入音樂譜例一



圖4.8 輸入音樂譜例二

　　但是，神經網路基本上是以音符之間的前後關係及機率來生成音樂，沒辦法對整首樂曲做架構的分析，因此較難做段落的區分或是主題變奏。圖4.9為一首機器生成的樂曲，我們可以發現其中並沒有很明確的樂句，整體的結構也比較鬆散。



圖4.9 機器生成樂曲樣本

4.3.2 樂曲獨特性及品質的兼顧

　　對一位作曲家來說，要創作出一首動聽的樂曲不僅需要具備完善的樂理及作曲知識，還要融合自己的巧思或是感情，要達到這樣的條件往往需要數年的功夫。而作曲所需的知識主要包括和聲學、對位法、曲式分析，這些知識都是透過前人大量的研究才整理出來的，當中的規則多變又繁複。因此，筆者認為要讓機器擁有作出與那些古典作曲家並駕齊驅的音樂作品，以現在的深度學習技術來說是有難度的。在絕大多數的情況，我們都會希望神經網路模型在訓練過程中的準確度接近100%，但對於作曲來說，我們希望曲子具有獨特性，而不是讓結果盡可能與輸入音樂相同，因此跟一般深度學習應用的領域有著不同的目標。另一方面，若要讓機器作出來的曲子有著較良好的品質，便需要將複雜多變的和聲對位技巧想辦法用機器可辨識的方式表示。但是，基本上若要把一首樂曲所有譜面訊息以及背後的所有和聲對位規則都標記，會需要大量的時間及背景知識，以現在的資料集來說，僅能從MIDI檔案提供的訊息來標記，但這些訊息只運用到最基本的樂理，其餘的訊息都須手動標記。

　　以本系統生成的樂曲來說，僅能在不考慮品質的情況下產生具獨特性的樂曲，而品質的部分就必須依靠運氣，有時可能會出現品質較優的段落，但可能其他段落的品質就需要改進。

4.4 系統特色及價值

　　現在類似電腦生成音樂的系統大多以爵士樂或是流行樂曲為主，幾乎沒有生成古典音樂的系統。而在生成古典音樂的系統中，與本系統較類似的有Feynman Liang 的 Bachbot[12]以及Gaetan Hadjeres的DeepBach[10]，這兩個系統同樣也是使用巴哈的聖詠合唱曲，不過他們比較著重於和聲重配，也就是在保持主旋律(第一聲部)的情況下，改變其他聲部的音型及音符配置，而本系統著重於生成全新獨特樂曲。

　　另外，由於巴哈的聖詠合唱曲每聲部皆為單一音符，上述的兩個系統僅針對此資料及提供單一音符的編碼，若聲部中有和弦便無法讀取。而本系統考慮到延展性，加入了和弦編碼的功能，如此一來，能夠讀取的樂曲種類便增加了許多。

第五章 結論與未來展望

5.1 結論

　　雖然目前深度學習的技術應用於音樂創作效果有限，但從這個專題當中，筆者學習到了不少深度學習的知識，也認知到深度學習還有哪些可以發展的空間以及需要改良的地方。

5.1.1 本系統待改進之處

　　1.樂曲架構不明確：本系統生成的樂曲並無明確的段落或是動機，一般的曲子通常可以分成幾個段落，而段落之間可能會有過門或是做為橋樑的銜接句，在一個段落之間會包含數個動機及樂句，這些動機可能會以不一樣的型式不斷出現於樂曲之中。

　　解決方法：針對資料集的音樂進行曲式分析，將段落、動機、樂句盡可能標記。

　　2.樂曲中存在不和諧的和聲：為了讓樂曲有獨特性，在機器生成音樂的過程中使用一些隨機的方法，因此，會產生一些不符合規則的和聲。

　　解決方法：事先針對和聲種類進行分類，並於輸入音樂中標記。在生成樂曲的過程中，先確定好主旋律之後再依照合理的和聲進行規則生成其他聲部。

　　3.主旋律會出現不符合規則的音程：在機器生成樂曲的主旋律中，我們可以發現常常會出現一些較奇怪的音程，導致旋律不太悅耳。以人類的角度，在寫一段旋律時，作曲家可能會融入自己的心情或是想表達的感受，他們會利用一些方法讓旋律聽起來好像正在表達一種心情，但是這些方法是透過人類的感受而想出來的，而機器對人類的情感毫無概念，因此較難在生成的樂曲中參雜這些元素。

　　解決方法：這個問題很難有較理想的解決方法，較可行的是將樂曲的旋律與人類的情感做歸類，在這當中找到一些規則，並於輸入音樂當中做標記，讓機器能夠試著學習這些規則。

5.1.2 從本專題學習到的知識

　　1.樂曲的和聲概念與架構：在思考如何解決機器生成音樂的一些問題時，筆者便從一些古典音樂中試著學習分析當中的和聲規則以及基本的曲式分析。

　　2.機器學習的基本概念：在接觸深度學習之前，筆者先從網路上瀏覽一些關於機器學習方法與步驟的文章，從中學習到基本的線性與邏輯迴歸分析、梯度下降法、資料正則化、代價函數．．．等。

　　3.不同種神經網路的差異及應用的領域：由於每種神經網路都有不同的特性以及能夠解決的問題，因此筆者針對能應用於音樂生成的神經網路進行學習，其中包括了LSTM、CNN、RBM。

5.2 未來展望

　　1.設計較複雜的神經網路模型：由於本系統的神經網路模型較簡易，因此訓練的效果相對較差，針對這方面改善或許能夠提高樂曲的品質。

　　2.增加輸入音樂的標記：目前的輸入資料僅包含音高、音符持續時間的訊息，未來會希望再增加標記種類，例如：力度、動機、和弦級數。

　　3.嘗試生成不同種風格的音樂：目前本系統是針對巴洛克時期的音樂設計，較不適合其他風格的輸入音樂，因此未來預計會針對不同風格的樂曲做資料表示法及編碼的改變。

參考文獻

1. 維基教科書 - 樂理與作曲
2. 古典音樂的發展. Available: <http://www3.tn.edu.tw/content/music/content/ext/classical/data-1-main01.htm>
3. Charloe Crawford, “An Introduction to Deep Learning”, 2016 , Available: <https://blog.algorithmia.com/introduction-to-deep-learning/>
4. Colah , “Understanding LSTM Networks”, 2015, Available:

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

1. Pranjal Srivastava, Essentials of Deep Learning: Introduction to Long Short Term Memory, 2017, Available:<https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/>
2. Numpy - Wikipedia, Available: <https://en.wikipedia.org/wiki/NumPy>
3. Numpy, Available: <http://www.numpy.org/>
4. Keras, Available: <https://keras.io/>
5. Music21, Available: <http://web.mit.edu/music21/>
6. Gaetan Hadjeres, Francois Pachet, Frank Nielsen, “DeepBach: a Steerable Model for Bach Chorales Generation”, 2015.
7. Dropout Layer - Introduction, Available: <https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html>
8. BachBot, Available: <http://bachbot.com/#/?_k=iz99kj>