Introduction

如何在現有的音樂類型、基礎上，生成獨特的音樂。各式類型的音樂都是由少數幾個音符透過排列、重組等方式而產生出，僅僅幾個音符就能製造出如此多種的音樂類型，於是我們想從音樂的基礎音符出發，來產出背景音樂。因我們這組有位組員平時會剪片上傳Youtube，但每次都要煩惱該用何種背景音樂，無版權的背景音樂大家都聽到膩了而有版權的又會沒收益，於是我們打算實際做一個能產生音樂的程式。

Literature Review/related Work

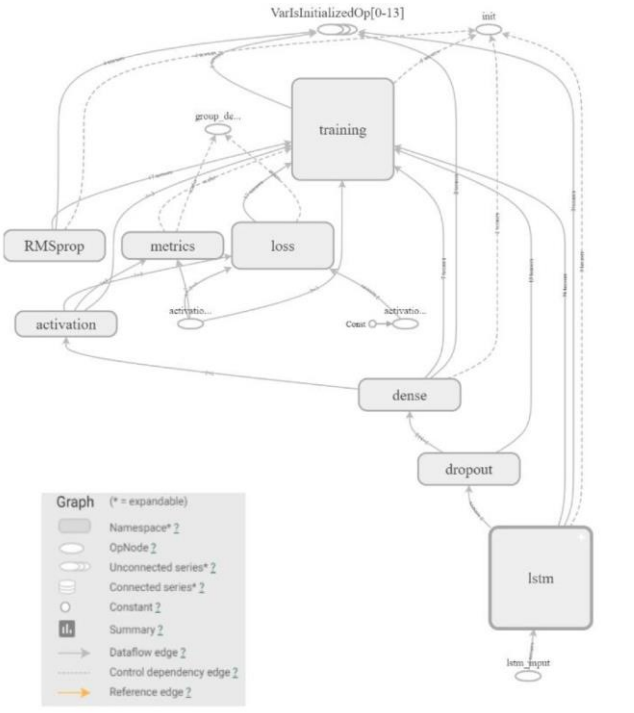
S.Mangal, R.Modak, P.Joshi **"LSTM Based Music Generation System"**

A. Deep Neural Network Design

將音符與出現時間點做連結，對音符與出現概率做分佈並使模型學習，利用該模型生成音符序列，Activation Layer決定了LSTM中哪些神經元需要被使用，或是該神經元的資訊在訓練模型時是否相干。LSTM其中的Dropout設為0.75。

B. SoftwareDesign

Note, Note Velocity, time interval of note構成了音符矩陣，其中，Note Velocity指的是音量大小，這個矩陣經過一些簡單的處理後，再丟入model中操作。



上圖為簡易的data ßow

我們使用了不一樣的資料處理方式，遍歷過所有的輸入後，將使用頻率過少的音符移除，能減少錯誤發生並加快模型訓練速度，取model的layer也採用了不同的堆疊方式及參數。除了LSTM之外，我　們還額外使用了其他兩種方式來生成音樂，分別是Conv1D與Conv1D和LSTM的結合。

Dataset

Input: MIDI files with some artists' piano music.(Basically from :http://www.piano-midi.de)
但下載的資源有些已損壞，於是我們手動移除。

Output:Melody generated by three model.

Baseline

相較於複雜的神經網路，使用隨機或有簡單規則的方法生成音符，在輸入的資料庫當中，使用出現頻率較高的音符作為輸出的原型。

Test Feasibility

Check_gpu.py這個檔案用來檢查gpu的資源是否可讓我們使用。

Data Fetch(Optional)

在loading_data.py這個檔案中，提供了另一種input data的方式，使用者輸入yt連結，該程式會將連結裡面的歌曲轉換成midi檔供主程式使用。

Main Approach

在主程式的起始，就給使用者兩個選項，從原有的資料庫中汲取資源抑或是從網路上抓取，這邊data一律統一用loading_data.py抓取。

```
#Choose your Dataset
option = int(input("Do you want to load dataset online? Please enter 1 if yes, otherwise 0 :"))

#Getting the list of notes as Corpus
all_midis= loading_data.capture_data(option)
Corpus= loading_data.extract_notes(all_midis)
print("Total notes in all the Chopin midis in the dataset:", len(Corpus))
print("First one hundred values in the Corpus:", Corpus[:100])
```

資料前處理

```
#Getting a list of rare chords & Eleminating them
rare_note = []
for index, (key, value) in enumerate(count_num.items()):
    if value < 10:
        m =  key
        rare_note.append(m)
print("Total number of notes that occur less than 10 times:", len(rare_note))

for element in Corpus:
    if element in rare_note:
        Corpus.remove(element)
print("Length of Corpus after elemination the rare notes:", len(Corpus))
```

將出現次數<10的音符移除，以利模型更快速的訓練並減少錯誤機率。

```
#Building dictionary to map an unique note to a number (ex:'E2': 115), and its reverse
mapping = dict((c, i) for i, c in enumerate(symbols))
reverse_mapping = dict((i, c) for i, c in enumerate(symbols))

print("Total number of notes:", corpus_length)
print("Number of unique notes:", symbol_length)
```

使音符以數字型態儲存，方便使用。

LSTM細節

我們的模型將依照以下順序進行建構：

1.LSTM Layer：具有512個unit，輸入型式為(X.shape[1],X.shape[2])，並做為下一層的輸入

2. Dropout Layer：有0.1的機率將輸入設為0，防止特殊特徵間有合作關係，使其用不完整的神經網路來學習。

3. LSTM Layer：具有256個unit，輸出向量。

4.Dense Layer：將輸入映射到輸出。

5.Dropout Layer：再次以0.1的機率使輸入設為0。

6.Dense Layer：具有y.shape[1]個unit，並使用softmax來進行多類別分類。

最後使用Adamax優化器來進行訓練。

這幾層能有效的使模型了解音符的結構信息，並減少overfit的風險。

Conv1D細節(在時間維度上應用卷積操作，用於捕捉序列數據中的局部特徵)

1.Conv1D Layer:擁有256個濾波器，kernel_size=3，並使用ReLU作為激活函數。

使用ReLU:非線性轉換使複數輸入轉換為0，計算效率較高，減輕梯度消失問題。

2.Dropout Layer:有0.1的機率使輸入設為0。

3.Dense Layer:將輸入映射到64維的輸出。

4.Dropout Layer:再一次以0.1的機率使輸入設為0。

5.Dense Layer:用於最後的分類操作，同樣使用softmax函數。

6.GlobalMaxPooling1D Layer:將序列維度的特徵壓縮為單一特徵，選擇每個特徵通道的最大值，提取最重要的特徵。

Merge Model細節

同時使用的LSTM與Conv1D的操作訓練模型。

```
    #Start training data
    model_LSTM.fit(X_train, y_train, batch_size=256, epochs=200)
    model_Conv1D.fit(X_train, y_train, batch_size=256, epochs=200)
    model_Merge.fit(X_train, y_train, batch_size=256, epochs=200)
```

將一樣的data丟入三個不同的model中做同樣次數的訓練。

下面以LSTM模型做輸出範例的解釋

```
197 def Malody_Generator_LSTM(Note_Count):
198     seed = X_seed[np.random.randint(0,len(X_seed)-1)]
199     Music = ""
200     Notes_Generated=[]
201     for i in range(Note_Count):
202         seed = seed.reshape(1,fearture_length,1)
203         prediction = model_LSTM.predict(seed, verbose=0)[0]
204         prediction = np.log(prediction) / 1.0 #diversity
205         exp_preds = np.exp(prediction)
206         prediction = exp_preds / np.sum(exp_preds)
207         index = np.argmax(prediction)
208         index_N = index/ float(symbol_length)
209         Notes_Generated.append(index)
210         Music = [reverse_mapping[char] for char in Notes_Generated]
211         seed = np.insert(seed[0],len(seed[0]),index_N)
212         seed = seed[1:]
213     #Now, we have music in form or a list of chords and notes and we want to be a midi file.
214     Melody = loading_data.chords_n_notes(Music)
215     Melody_midi = stream.Stream(Melody)
216     return Music,Melody_midi
217
218 Music_notes_LSTM, Melody_LSTM = Malody_Generator_LSTM(300)
219 Melody_LSTM.write('midi','LSTM.mid')
```

197 Note_Count:欲生成的音符數量
198 seed:隨機從X_seed中選取一個當作生成起點
203:用LSTM模型對種子序列做預測，以獲取下個音符的概率分佈
204/205:進行對數轉換與指數轉換，用以增加生成音樂的豐富性
206:正規化預測結果，使其總和為1
207/208:找到概率分佈的最大值，作為生成音符參考
210:將型態轉換為實際音符
211/212:在序列末端插入生成的音符，並移除第一個元素，為下次生成做準備
213:轉換為樂譜型態
214使其方便寫入midi檔

最後，將不同模型產生的midi檔存入相對應的檔案名稱。

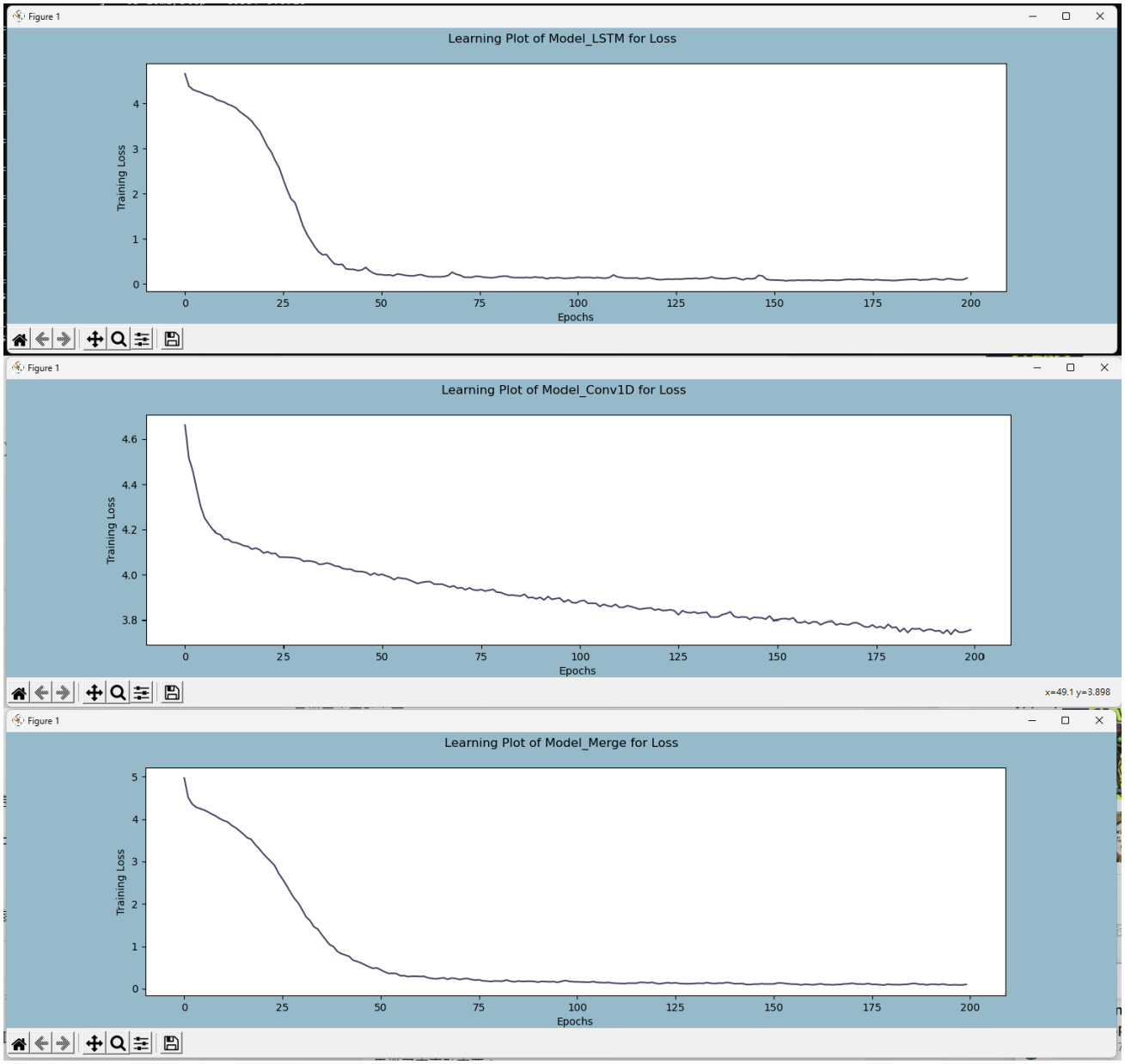
Evaluation Matrix

Categorical_crossentropy:

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

y_i True_label　　$\hat{y}_i$ Pred_label

根據true_label與pred_label的差距，差距越小代表模型能更精準地預測各個類別的概率分佈。



Result and Analysis

輸出的音檔在我們聽起來都是差不多的，可能是我們對音樂的鑑賞性不好，或是輸出的音樂鑑別度不佳，但至少有輸出東西來。但音樂就是音樂，並不會有什麼規則須依循，因此我們對輸出的結果沒什麼太大的問題，但要如何產出某特定類型或風格的音樂，這可能就需要修改程式架構或增加功能等才能達成。針對輸出的結果，我們沒辦法對此程式有什麼評價，只能由evaluation matrix使用的function來計算loss值，透過數據來評判這個model，但數據與真實輸出卻感覺也沒什麼太大的連　結，畢竟我們原本就只想生成規避版權且與原有不同的音樂。

Error Analysis

生成的音檔鑑別度不高且輸入需具有一定的規格，難以使用多種不同的音樂當作輸入。

生成的音樂只會依照單一節奏(ex.BPM無法更改)，略顯單調。

Future Work

在模型的訓練上參考更多其他paper，調整layer架構且找到更好的參數設定，並試著使輸出能依照　使用者的需求更改類型。

使生成的音樂能以不同的節奏呈現，增加變化。

未來想嘗試使用AI生成嘻哈歌曲的beat，輸入的資源可能就要以嘻哈歌曲為主體，單一節奏的問題　也須修改，增加多樣性以朝AI beat maker為目標邁進。

Future Issue

畢竟我們的輸入是使用其他的製作的歌曲，輸出的結果可能會有侵權的疑慮產生，能否開啟收益這部分有待檢閱過相關法律條文或yt規範後再來進一步討論。

Code Link

https://github.com/chiafu2018/AI_generate_music

Reference

1.Yang, L.C., A.Lerch, :On the evaluation of generative models in music

2. S.Mangal, R.Modak, P.Joshi :LSTM Based Music Generation System