

# 靜宜大學

## 資訊工程學系

### 畢業專題成果報告書

#### AI音樂生成器

學生：

資工四A    411003992 方芊嵐

資工四A    411018206 曾歆玲

指導教授：林耀鈴 教授

西元二〇二四年十二月

# 目錄

一、前言.....	3
二、研究目的.....	3
三、文獻探討.....	4
1.生成式 AI 技術.....	4
2.音樂生成的技術框架.....	4
3.ABC 記譜法的特性與應用場景.....	4
4.深度學習在音樂生成中的應用挑戰.....	4
四、研究方法.....	5
1. 專題簡介.....	5
2. 開發環境.....	5
3. 系統設計與方法概述.....	5
3.1總體架構.....	5
3.2資料讀取.....	5
3.3 數據預處理.....	6
3.4資料分批.....	7
3.5模型設計.....	7
3.6模型訓練.....	8
3.7文本生成.....	8
3.8結果保存.....	9
4. 實驗結果.....	10
五、結果與討論.....	13
六、參考文獻.....	14

## 一、前言

音樂創作是一種極具創造力的活動，而基於人工智慧的音樂生成技術為創作者提供了一種全新的輔助工具。近年來，深度學習技術的快速發展使遞歸神經網路 (Recurrent Neural Networks, RNN) 在處理序列數據 (如音樂、文本) 方面展現了強大的潛力。隨著人工智慧技術的快速發展，生成式 AI 在音樂領域的應用愈發廣泛。音樂作為一種高度結構化且富有表現力的創作形式，其生成具有挑戰性。傳統上，音樂創作主要依賴於作曲家的靈感與專業知識，而生成式 AI 的引入，為音樂創作提供了一種全新的協作方式。AI 音樂生成器利用深度學習技術自動生成旋律和和聲，創造出多樣化的音樂片段，為創作者和音樂愛好者提供靈感，並幫助非專業人士體驗音樂創作的樂趣。AI 音樂生成技術不僅可以在娛樂、遊戲音效等應用場景中提供背景音樂，還能作為教育工具，幫助學習音樂創作的理論知識。此外，音樂生成器也適合於靈感啟發、音樂療法等新興領域。基於此，我們利用生成式 AI 模型生成音樂，透過 AI 自動化生成旋律、和聲和不同風格的音樂片段，為創作者提供靈感、探索 AI 在音樂創作中的應用潛力。

## 二、研究目的

本研究的目的是開發一個基於 AI 的音樂生成器，利用生成式模型學習和模仿音樂結構，實現不同風格和長度的音樂生成，通過 LSTM 模型學習音樂序列的特徵，生成不同風格的旋律片段。具體而言，本研究目標包括以下幾點：

1. 探索音樂序列生成技術  
利用遞歸神經網路 (RNN) 及其變體 LSTM 的特性，分析其在音樂序列生成中的應用效果，驗證其能否學習並模擬音樂數據的時間依賴性和結構性。
2. 提升音樂創作效率  
通過自動化技術，輔助音樂創作者生成具有創意的音樂片段，減少手動編曲的重複性勞動。
3. 提供自動化音樂工具  
開發一個支持即時生成、播放與保存的工具，便於創作者直接利用生成的音樂進行二次創作或即時應用。
4. 擴展 ABC 記譜法應用  
探索標準化音樂記譜法 (如 ABC 記譜法) 的應用可能性，驗證其作為數據表示格式的適用性，並分析其在生成模型中的表現。

### 三、文獻探討

#### 1. 生成式 AI 技術

生成式 AI 是通過深度學習模型生成新內容的技術，常見模型包括 LSTM、GAN 和 Transformer。LSTM (Long Short-Term Memory) 適合處理時序數據，特別適用於捕捉音符之間的長期依賴性，因而成為音樂生成的主要技術之一。Eck 和 Schmidhuber (2002) 早期將 LSTM 應用於藍調音樂的即興創作，證明了 LSTM 在生成音樂序列方面的有效性。此外，Transformer 模型在近年興起，因其優秀的並行處理能力和長期依賴性捕捉能力，被應用於如 OpenAI 的 MuseNet 等音樂生成系統中。

#### 2. 音樂生成的技術框架

音樂生成技術涉及數據處理、模型訓練和音樂序列生成三個步驟。Magenta 是 Google 開發的生成式 AI 音樂創作工具，其 PerformanceRNN 和 MusicVAE 框架能夠生成多樣化的旋律片段。PerformanceRNN 採用了 LSTM 網絡，專注於生成即時表演性強的音樂；而 MusicVAE 則通過變分自編碼器 (VAE) 生成多樣性更高的旋律，擅長處理不同風格之間的轉換。此外，OpenAI Jukebox 結合 Transformer 模型進行多樂器音樂的生成，進一步提高了生成音樂的質量和豐富性。

#### 3. ABC 記譜法的特性與應用場景

由 Walshaw (1993) 開發的 ABC 記譜法是一種基於純文本的音樂記譜格式，使用簡單的字母和符號來表示音樂片段 (如音符、高度和節奏)。這種記譜法因其輕量級、可讀性高且易於解析的特性，被廣泛應用於音樂研究和生成工具中。優勢：1. 能夠通過簡單的文本處理技術輕鬆生成和修改樂譜。2. 與現代深度學習模型結合時，能有效轉換為序列數據供模型訓練，降低數據預處理的複雜性。應用案例：Google 的 Magenta 和其他音樂生成項目中使用了類似格式來表示 MIDI 數據，驗證了文本化音樂表示在人工智慧應用中的價值。

#### 4. 深度學習在音樂生成中的應用挑戰

Briot 等人 (2020) 指出，深度學習生成音樂時會遇到一些挑戰，例如旋律連貫性和多樣性的平衡、風格模仿的準確性和模型運行的高效性。模型往往需要大量音樂數據才能生成符合人類音樂審美的旋律，這對數據集的多樣性和模型的訓練效果提出了高要求。Herremans 和 Chuan (2017) 則提到，音樂生成的未來發展方向在於多樂器和多風格的結合，期望生成器能夠生成更接近人類創作的音樂。

## 四、研究方法

---

### 1. 專題簡介

我們的專題旨在基於深度學習模型(LSTM 網路)進行音樂生成。藉助 TensorFlow 和 MIT 的深度學習工具包, 我們訓練了一個遞歸神經網路(RNN), 能夠生成結構化的音樂片段(ABC 格式), 並提供即時檢查、播放和存檔功能。

---

### 2. 開發環境

- 框架: TensorFlow 2.x、MIT Deep Learning Package
  - 語言: Python
  - 硬體: GPU 加速器
  - 依賴項目:
    - `numpy`, `os`, `time`
    - `tqdm`: 進度條顯示
    - `IPython.display`: 即時音樂播放
    - `abcmidi` 和 `timidity` (ABC 音樂處理工具)
- 

### 3. 系統設計與方法概述

#### 3.1 總體架構

整個系統分為以下四個模塊:

1. 數據處理模組: 從 ABC 文件中讀取音樂數據, 進行編碼和向量化處理。
2. 模型構建模組: 設計 LSTM 模型結構, 並設定訓練流程與損失函數。
3. 音樂生成模組: 根據用戶輸入的起始字符串生成新的音樂片段。
4. 結果處理模組: 將生成的音樂片段保存為文件。

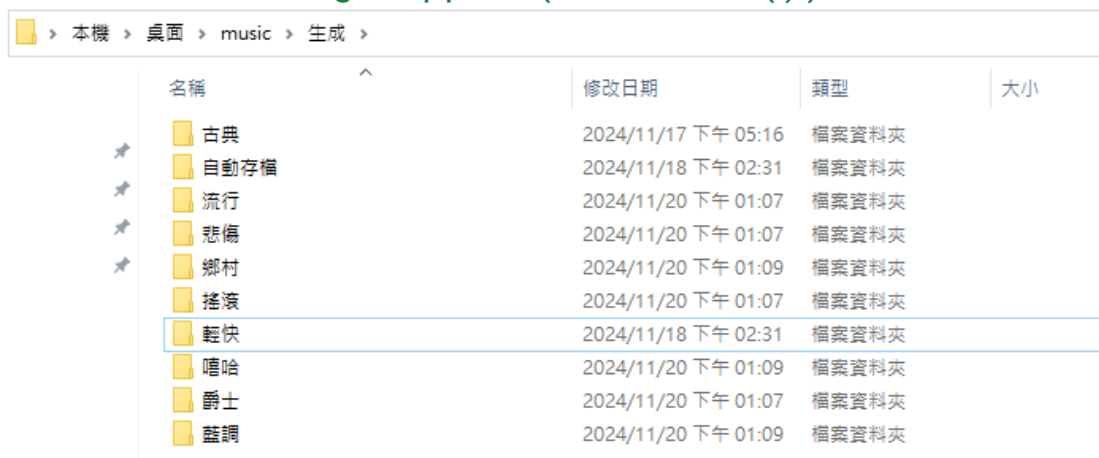
## 3.2 資料讀取

功能

- 讀取存放於指定資料夾中的 **.abc** 音樂文件。
- 將所有音樂文件內容匯總為列表 **songs**, 並計算其總數。

程式碼片段

```
python
folder_path = 'C:\\\\Users\\user\\Desktop\\music\\'
+ temp
for filename in os.listdir(folder_path):
    if filename.endswith('.abc'):
        with open(os.path.join(folder_path,
filename), 'r') as file:
            songs.append(file.read())
```



名稱	修改日期	類型	大小
古典	2024/11/17 下午 05:16	檔案資料夾	
自動存檔	2024/11/18 下午 02:31	檔案資料夾	
流行	2024/11/20 下午 01:07	檔案資料夾	
悲傷	2024/11/20 下午 01:07	檔案資料夾	
鄉村	2024/11/20 下午 01:09	檔案資料夾	
搖滾	2024/11/20 下午 01:07	檔案資料夾	
輕快	2024/11/18 下午 02:31	檔案資料夾	
嘻哈	2024/11/20 下午 01:09	檔案資料夾	
爵士	2024/11/20 下午 01:07	檔案資料夾	
藍調	2024/11/20 下午 01:09	檔案資料夾	

## 3.3 數據預處理

步驟

1. 將所有音樂文本合併為單一字符串。
2. 建立字符與索引的對應關係：
  - **char2idx**: 將字符映射為數字。
  - **idx2char**: 將數字映射回字符。
3. 將文本轉換為數字向量。

程式碼片段

```
python
char2idx = {u: i for i, u in enumerate(vocab)}
vectorized_songs = np.array([char2idx[char] for
char in songs_joined])
```

### 3.4 資料分批

功能

- 將數字向量劃分為多組固定長度的輸入與標籤。
- 輸入為前 `seq_length` 的字符，標籤為後 `seq_length` 的字符。

程式碼片段

python

```
def get_batch(vectorized_songs, seq_length,
batch_size):
    idx = np.random.choice(len(vectorized_songs)
- seq_length, batch_size)
    input_batch = [vectorized_songs[i:i +
seq_length] for i in idx]
    output_batch = [vectorized_songs[i + 1:i +
seq_length + 1] for i in idx]
    return np.reshape(input_batch, [batch_size,
seq_length]), np.reshape(output_batch,
[batch_size, seq_length])
```

### 3.5 模型設計

架構

1. 嵌入層 (Embedding) : 將字符索引映射為密集向量。
2. LSTM 層 : 處理序列數據並提取時間特徵。
3. 全連接層 (Dense) : 將 LSTM 的輸出映射回字符空間。

程式碼片段

python

```
def build_model(vocab_size, embedding_dim,
rnn_units, batch_size):
    return tf.keras.Sequential([
        tf.keras.layers.Embedding(vocab_size,
embedding_dim, batch_input_shape=[batch_size,
None]),
        tf.keras.layers.LSTM(rnn_units,
```

```
return_sequences=True, stateful=True,
recurrent_initializer='glorot_uniform'),
        tf.keras.layers.Dense(vocab_size)
    ])
```

### 3.6 模型訓練

損失函數

- 使用交叉熵損失函數計算預測值與實際標籤之間的差異。

優化器

- 使用 Adam 優化器, 學習率為  $5e-3$ 。

訓練步驟

1. 加載數據並進行前向傳播。
2. 計算損失並執行反向傳播以更新模型參數。
3. 保存模型權重。

程式碼片段

python

```
@tf.function
def train_step(x, y):
    with tf.GradientTape() as tape:
        y_hat = model(x)
        loss = compute_loss(y, y_hat)
        grads = tape.gradient(loss,
model.trainable_variables)
        optimizer.apply_gradients(zip(grads,
model.trainable_variables))
    return loss
```

### 3.7 文本生成

步驟

1. 將起始字符串轉換為數字向量。
2. 循環生成 `generation_length` 長度的字符, 並將每個預測結果作為下一次輸入。



### 3. 返回生成的完整文本。

程式碼片段

```
python
def generate_text(model, start_string,
generation_length=1000):
    input_eval = [char2idx[s] for s in
start_string]
    input_eval = tf.expand_dims(input_eval, 0)
    text_generated = []
    model.reset_states()
    for i in range(generation_length):
        predictions = model(input_eval)
        predictions = tf.squeeze(predictions, 0)
        predicted_id =
tf.random.categorical(predictions,
num_samples=1)[-1, 0].numpy()
        input_eval =
tf.expand_dims([predicted_id], 0)

    text_generated.append(idx2char[predicted_id])
    return (start_string +
''.join(text_generated))
```


### 3.8 結果保存

功能

- 將每首生成的音樂片段保存為 **.abc** 文件。
- 同時將所有片段存儲到一個匯總文本文件中。

程式碼片段

```
python
file_path = f"C:\\Users\\user\\Desktop\\music\\生
成\\{temp}\\{i}_{timestamp}.abc"
with open(file_path, 'w', encoding='utf-8') as f:
    f.write(song)
```

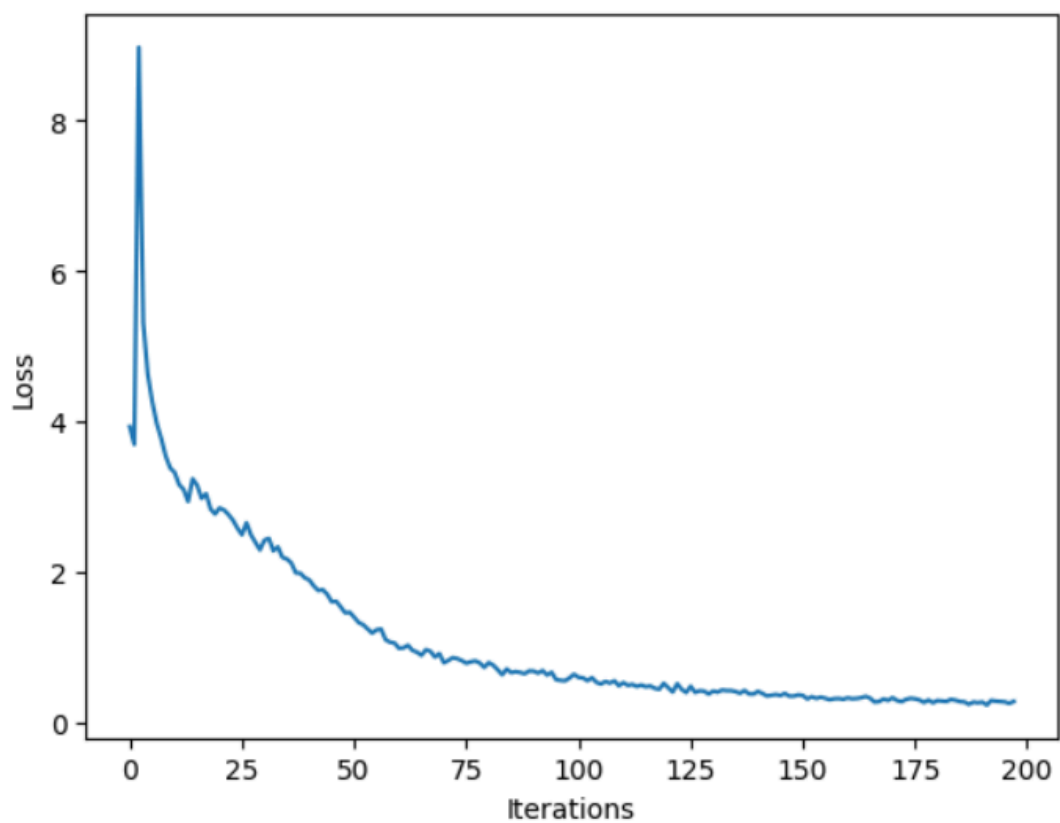
<div>  <span>本機 &gt; 桌面 &gt; music &gt; 生成 &gt; 自動存檔</span> </div>				
	名稱	修改日期	類型	大小
★	generated_songs_輕快_20241118_143...	2024/11/18 下午 02:31	文字文件	2 KB
★	generated_songs_輕快_20241118_143...	2024/11/18 下午 02:31	文字文件	2 KB
★	generated_songs_輕快_20241118_143...	2024/11/18 下午 02:30	文字文件	2 KB
★	generated_songs_輕快_20241118_142...	2024/11/18 下午 02:29	文字文件	2 KB
	generated_songs_輕快_20241117_183...	2024/11/18 下午 02:28	文字文件	1 KB
	generated_songs_輕快_20241118_142...	2024/11/18 下午 02:28	文字文件	2 KB
	generated_songs_輕快_20241117_183...	2024/11/17 下午 06:34	文字文件	1 KB
	generated_songs_輕快_20241117_183...	2024/11/17 下午 06:33	文字文件	1 KB
	generated_songs_輕快_20241117_183...	2024/11/17 下午 06:33	文字文件	1 KB
	generated_songs_輕快_20241117_183...	2024/11/17 下午 06:33	文字文件	2 KB
	generated_songs_輕快_20241117_182...	2024/11/17 下午 06:29	文字文件	2 KB
	generated_songs_輕快_20241117_182...	2024/11/17 下午 06:28	文字文件	2 KB
	generated_songs_輕快_20241117_182...	2024/11/17 下午 06:28	文字文件	2 KB
	generated_songs_輕快_20241117_182...	2024/11/17 下午 06:28	文字文件	2 KB
	generated_songs_輕快_20241117_182...	2024/11/17 下午 06:27	文字文件	2 KB
	generated_songs_輕快_20241117_181...	2024/11/17 下午 06:17	文字文件	2 KB
(C:)	generated_songs_古典_20241117_180...	2024/11/17 下午 06:09	文字文件	0 KB
	generated_songs_古典_20241117_180...	2024/11/17 下午 06:08	文字文件	0 KB
	generated_songs_古典_20241117_180...	2024/11/17 下午 06:08	文字文件	0 KB
	generated_songs_古典_20241117_180...	2024/11/17 下午 06:08	文字文件	0 KB
	generated_songs_輕快_20241117_180...	2024/11/17 下午 06:01	文字文件	2 KB
	generated_songs_輕快_20241117_175...	2024/11/17 下午 05:59	文字文件	2 KB
	generated_songs_古典_20241117_175...	2024/11/17 下午 05:54	文字文件	3 KB
	generated_songs_古典_20241117_175...	2024/11/17 下午 05:51	文字文件	2 KB
	generated_songs_古典_20241117_174...	2024/11/17 下午 05:48	文字文件	3 KB
	generated_songs_古典_20241117_174...	2024/11/17 下午 05:43	文字文件	1 KB
	generated_songs_古典_20241117_173...	2024/11/17 下午 05:36	文字文件	1 KB
	generated_songs_古典_20241117_173...	2024/11/17 下午 05:35	文字文件	3 KB
	generated_songs_古典_20241117_173...	2024/11/17 下午 05:34	文字文件	5 KB
	generated_songs_古典_20241117_173...	2024/11/17 下午 05:30	文字文件	4 KB
	generated_songs_古典_20241117_173...	2024/11/17 下午 05:30	文字文件	4 KB
	generated_songs_古典_20241117_172...	2024/11/17 下午 05:27	文字文件	3 KB
	generated_songs_古典_1_20241117_1...	2024/11/17 下午 05:22	文字文件	3 KB

## 4. 實驗結果

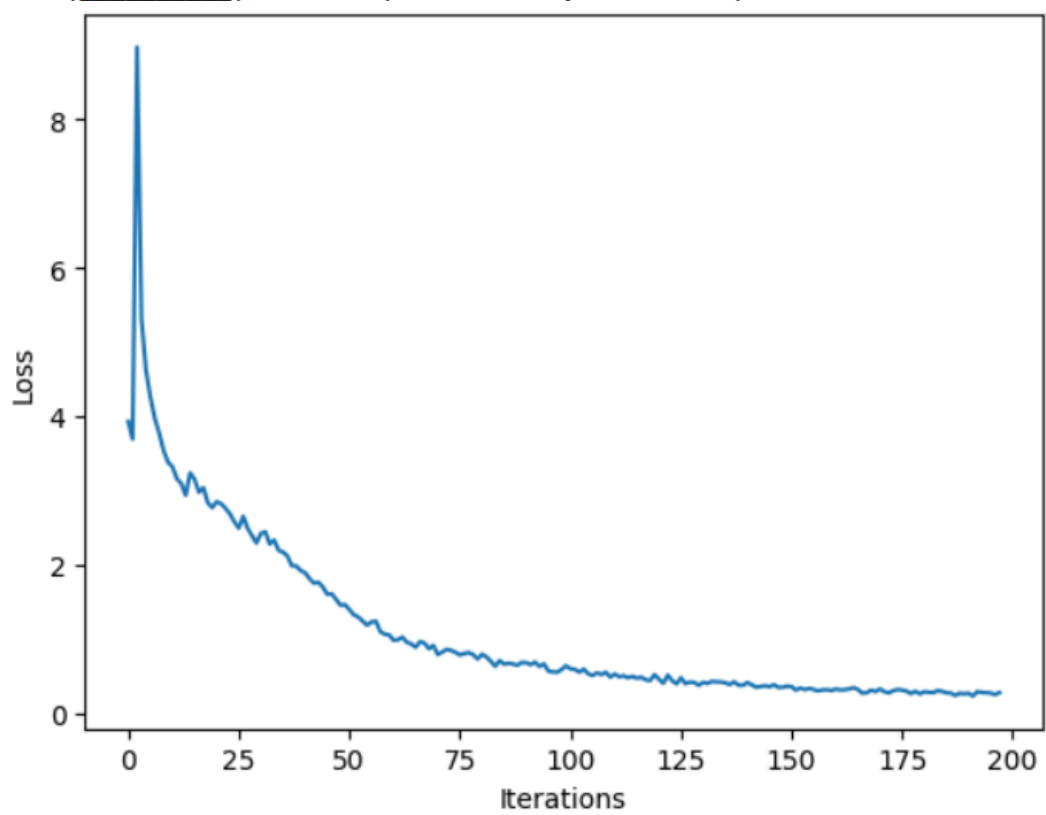
### 訓練過程

模型通過 200 次迭代進行訓練，損失逐漸下降，模型能夠生成合理的 ABC 音樂片段。

### 生成效果



100% |██████████| 200/200 [02:08<00:00, 1.56it/s]



生成的音樂格式正確, 部分片段可運用。

輸出示例：

```
makefile
```

```
X:4
```

```
T:Happy Jig
```

```
M:9/8
```

```
L:1/8
```

```
K:C
```

```
CDE GEC | GAB cde | fed cBA | G3 G3 ||
```

再將以上的abc樂譜轉成可撥放的mp3檔

```
song = ""
```

```
X:4
```

```
T:Happy Jig
```

```
M:9/8
```

```
L:1/8
```

```
K:C
```

```
CDE GEC | GAB cde | fed cBA | G3 G3 ||
```

```
""
```

```
print("\nsong: ")
```

```
print(song)
```

```
# Convert the ABC notation to audio file and listen to it
```

```
mdl.lab1.play_song(song)
```

song:

```
X:4
```

```
T:Happy Jig
```

```
M:9/8
```

```
L:1/8
```

```
K:C
```

```
CDE GEC | GAB cde | fed cBA | G3 G3 ||
```

▶ 0:07 / 0:07 ———— 🔊 ⋮

使用HackMD將abc譜轉成五線譜

The screenshot shows the HackMD web editor. The top bar includes the HackMD logo, a search icon, and a toolbar with icons for undo, redo, bold, italic, link, unlink, list, and other editing functions. The main text area contains the following ABC notation:

```
1  `` `abc
2  X:4
3  T:Happy Jig
4  M:9/8
5  L:1/8
6  K:C
7  CDE GEC | GAB cde | fed cBA | G3 G3 ||
8  `` `
```

Below the text area, there is a section titled "Happy Jig" showing the musical score in staff notation. The score is in treble clef, 9/8 time, and C major. It consists of two measures of music.

## 五、結果與討論

### 1. 優化與未來方向

1. 優化超參數:調整 LSTM 單元數量、序列長度等。
2. 模型改進:嘗試其他類型的 RNN 或 Transformer 架構。
3. 數據增強:增加更多樣化的音樂片段進行訓練。
4. 生成控制:增加用戶參數設置功能(如節拍、調性)以生成指定風格的音樂。

### 2. 結論

本專題探討基於深度學習的音樂生成技術,利用遞歸神經網路(LSTM)學習音樂序列的長期依賴關係,並結合 ABC 記譜法的文本化特性,成功構建了一個能生成特定風格音樂的自動化系統。實驗證明,該模型能有效捕捉旋律規律和和聲邏輯,生成具有創意性且結構合理的音樂片段。同時,ABC 記譜法的簡單性與可擴展性為數據處理和存儲帶來了便利。然而,生成結果的多樣性和平衡性仍面臨挑戰,特別是在風格細化與質量量化評估方面。未來研

究可著重於多風格數據集的擴展、實時生成技術的應用，以及與音樂編輯工具的結合，實現更為實用的互動式創作應用。通過進一步優化，我們可以生成更多樣化且更具藝術性的音樂片段。

## 六、參考文獻

### 1. 生成音樂技術和模型

- Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., et al. (2018). *Music Transformer: Generating Music with Long-Term Structure*. 本文介紹了使用 Transformer 模型生成音樂的技術，強調了如何處理長期依賴性。
- Eck, D., & Schmidhuber, J. (2002). *Finding temporal structure in music: Blues improvisation with LSTM recurrent networks*. 這是關於使用 LSTM 模型來生成音樂的早期研究，特別是在即興音樂方面的應用。
- Hadjeres, G., Pachet, F., & Nielsen, F. (2017). *DeepBach: a Steerable Model for Bach Chorales Generation*. 本研究展示了如何使用生成模型來模擬巴赫風格的合唱曲，提供了有關音樂風格模仿的詳細技術。

### 2. 音樂生成的工具和框架

- Magenta Project (TensorFlow). 由 Google 開發的開源研究項目，專注於探索如何將機器學習應用於音樂和藝術創作。提供了多種工具和模型，如 MusicVAE 和 PerformanceRNN。
- OpenAI Jukebox. 開源神經網絡模型，能夠生成具有歌詞和伴奏的多風格音樂。文獻和模型解釋可參考 [OpenAI Jukebox 官方博客](#)。

### 3. MIDI 和音樂數據處理

- Raffel, C. (2016). *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*. 此文獻討論了 MIDI 文件如何處理與音樂序列的比對和對齊技術，對 MIDI 數據預處理非常有幫助。

### 4. 深度學習與音樂生成的綜述

- Briot, J.-P., Hadjeres, G., & Pachet, F.-D. (2020). *Deep Learning Techniques for Music Generation – A Survey*. 這篇文獻提供了有關音樂生成的深度學習技術的全面回顧，涵蓋了各種模型和方法的優缺點。
- Herremans, D., Chuan, C.-H., & Chew, E. (2017). *A Review of Deep Learning Techniques for Music Generation*. 一篇綜述，介紹了如何使用深度學習技術生成音樂，涵蓋不同的網絡架構及其在音樂生成中的應用。

### 5. 其他有用的資源

- PrettyMIDI Library. 用於 MIDI 文件的處理和轉換的 Python 庫，對於將音符序列轉換為 MIDI 以及分析音樂結構很有幫助。
- MuseNet by OpenAI. 一種基於 Transformer 的模型，能生成多樂器的音樂，並支持多風格切換。