

靜宜大學

資訊工程學系

畢業專題成果報告書

AI 音樂生成

學生：

資工四 A 411147615 張詰沅

資工四 A 411147576 吳松翰

資工四 A 411147437 余承恩

資工四 A 411134507 張晨瑋

指導教授：林耀鈴 教授

西元二〇二五年十二月

靜宜大學 資訊工程學系 A I 音樂生成 西元二〇二五年十二月

學生：張詰沅、吳松翰、余承恩、張晟瑋

指導教授：林耀鈴

靜宜大學資訊工程學系

摘要

本專題旨在研究與實作一套以深度學習為核心的 AI 音樂生成系統，探討 Transformer 模型於 MIDI 旋律生成上的可行性與應用潛力。隨著生成式人工智慧技術快速發展，音樂生成逐漸成為跨領域研究的重要主題，而 MIDI 資料具備事件化、結構清晰與資訊完整等特性，適合作為模型之訓練來源。本專題以公開 MIDI 曲庫為資料基礎，透過資料清理、節奏量化、音域修正、事件序列轉換與移調資料增強，建立一致化且可供模型學習的大型資料集。

在模型設計方面，本研究採用事件序列（Event-based Representation）作為音樂表示方式，並以 Transformer 架構作為模型主體，使其能透過自注意力機制有效捕捉旋律與節奏的長距依賴關係。模型訓練流程包含資料 Token 化、序列切片、批次建構、損失監控與最佳化策略調整。在生成階段，系統採用自回歸方式逐步產生音樂事件，並結合多項生成控制機制，例如 top-k、top-p 抽樣、temperature、音域限制、最短音長、最大同時音數與調性約束，以提升旋律流暢度與音樂性。

研究結果顯示，模型能生成具備基本樂句結構與節奏邏輯之旋律片段，並於不同參數設定下呈現多樣化的音樂風格特徵。搭配後處理與規則約束，生成結果在音域合理性、節奏一致性與可聆聽性方面皆有明顯改善。此外，本專題亦整合簡易互動式介面，使用者可直接調整生成參數並播放輸出結果，提高系統的可操作性與展示價值。

整體而言，本專題成功驗證 Transformer 模型於 MIDI 音樂生成的實作可行性，並建構了一套能自動生成旋律草稿之 AI 音樂生成系統，具備作曲輔助、教學應用與創意啟發的潛力。未來可進一步擴展至多軌和聲生成、風格控制、長篇曲式結構規劃及音訊層級生成等方向，以提升系統的完整性與實用性。

靜宜大學資訊工程學系
專題實作授權同意書

本人具有著作財產權之論文全文資料，授予靜宜大學資工系，為學術研究之目的以各種方法重製，或為上述目的再授權他人以各種方法重製，不限地域與時間，惟每人以一份為限。授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。

指導教授 林耀鈴

| | | | |
|-------|-----|--------------|-----------------------|
| 學生簽名: | 張詰沅 | 學號:411147615 | 日期:西元 2025 年 12 月 3 日 |
| 學生簽名: | 吳松翰 | 學號:411147576 | 日期:西元 2025 年 12 月 3 日 |
| 學生簽名: | 余承恩 | 學號:411147437 | 日期:西元 2025 年 12 月 3 日 |
| 學生簽名: | 張晨瑋 | 學號:411134507 | 日期:西元 2025 年 12 月 3 日 |
| 學生簽名: | | 學號: | 日期:西元 年 月 日 |

指導教師簽章 _____

西元 2025 年 12 月 3 日

指導教師確認書

靜宜大學資訊工程學系
專題實作指導教師確認書

茲確認專題書面報告之格式及內容符合本系之規範

畢業專題實作名稱 : _____AI 音樂生成_____

畢業專題實作分組名單 : 共計 4 人

| 組員姓名 | 學號 |
|------|-----------|
| 張詰沅 | 411147615 |
| 吳松翰 | 411147576 |
| 余承恩 | 411147437 |
| 張晨瑋 | 411134507 |
| | |
| | |

指導教師簽章 _____
西元 2025 年 12 月 3 日

誌謝

本專題得以順利完成，首先要由衷感謝指導老師 **林耀鈴 老師** 在研究過程中所提供的專業指導與寶貴建議。林老師不僅在技術上給予許多精確的方向，更在研究思考與實作方法上提供耐心的引導，使本專題得以逐步完善。

同時，也感謝在專題期間協助討論、提供意見與支援的同學與朋友，使研究過程更加順利。此外，亦感謝家人在本專題執行期間所給予的理解與鼓勵，讓我能無後顧之憂地投入研究工作。

最後，向所有曾在此過程中提供協助與支持的人士致上誠摯的謝意。謹此致謝。

目 錄

| | | |
|-------|--------------------------------|-----|
| 中文摘要 | | iii |
| 誌謝 | | vi |
| 目錄 | | vii |
| 圖目錄 | | ix |
| 第一章、 | 緒論..... | 1 |
| 1.1 | 研究背景..... | 1 |
| 第二章、 | 專題內容與進行方法..... | 2 |
| 2.1 | 研究動機..... | 2 |
| 2.2 | 研究目的..... | 2 |
| 2.3 | 系統架構概述..... | 3 |
| 2.4 | 資料來源與資料處理方法..... | 3 |
| 2.4.1 | 資料來源..... | 3 |
| 2.4.2 | 事件序列 編碼方式..... | 3 |
| 2.5 | Music Transformer 模型架構..... | 4 |
| 2.6 | 模型訓練流程..... | 4 |
| 2.7 | 音樂生成方法..... | 5 |
| 2.8 | 系統開發工具與環境..... | 5 |
| 第三章、 | 專題流程與架構..... | 6 |
| 3.1 | 系統整體架構..... | 6 |
| 3.2 | 系統流程圖..... | 7 |
| 3.3 | 各模組功能與設計理念..... | 9 |
| 3.3.1 | 資料來源模組..... | 9 |
| 3.3.2 | MIDI 前處理模組..... | 9 |
| 3.3.3 | 事件序列編碼模組..... | 10 |
| 3.3.4 | Music Transformer 訓練模組..... | 10 |
| 3.3.5 | 音樂生成模組..... | 11 |
| 3.4 | 系統優點..... | 11 |
| 3.5 | 本章小結..... | 11 |
| 第四章、 | 專題成果介紹..... | 12 |
| 4.1 | 模型訓練成果..... | 12 |
| 4.2 | 生成樂曲成果..... | 13 |
| 4.3 | 系統介面與操作成果..... | 14 |
| 4.4 | 本章小結..... | 15 |
| 第五章、 | 專題學習歷程介紹..... | 16 |
| 5.1 | 專題相關軟體學習介紹..... | 16 |
| 5.1.1 | Python 與 Google Colab 之應用..... | 16 |

| | | |
|-------------|-----------------------------------|-----------|
| 5.1.2 | MIDI 處理與事件序列相關套件..... | 17 |
| 5.1.3 | 深度學習框架與 Music Transformer 架構..... | 18 |
| 5.1.4 | 音樂播放與視覺化工具..... | 18 |
| 5.2 | 專題製作過程遭遇的問題與解決方法..... | 19 |
| 5.2.1 | 資料與事件序列設計上的問題..... | 19 |
| 5.2.2 | 模型訓練與超參數調整的問題..... | 20 |
| 5.2.3 | 生成階段的實務問題與修正..... | 21 |
| 5.2.4 | 介面與使用體驗相關問題..... | 22 |
| 5.3 | 本章小結..... | 22 |
| 第六章、 | 結論與未來展望..... | 23 |
| 6.1 | 研究結論..... | 23 |
| 6.2 | 研究限制..... | 24 |
| 6.3 | 未來展望..... | 25 |
| 6.4 | 專題心得與自我成長..... | 26 |
| 6.5 | 本章小結..... | 26 |
| 參考文獻 | | 27 |

圖 目 錄

| | | |
|--------|----------------------|----|
| 圖 4-1、 | 模型訓練曲線..... | 12 |
| 圖 4-1、 | 系統主畫面及生成設定與輸出畫面..... | 14 |

第一章 緒論

1.1 研究背景

隨著深度學習在語音與影像生成領域的快速發展，人工智慧在音樂生成上的應用也逐漸成熟。傳統 AI 音樂生成大多依賴馬可夫鏈或基於規則的演算法，生成結果在旋律延展性與音樂結構上皆存在侷限。近年來，Transformer 架構因其優異的序列建模能力，在自然語言處理與聲音生成上展現出高度潛力。其中，Google Magenta 提出的 Music Transformer 透過相對位置編碼 (Relative Positional Encoding) 與事件序列(Event-based Representation) 的方式，使模型能夠有效理解樂句的前後關係，進而生成更具音樂性、結構性與表現力的樂曲。

在此背景下，本專題旨在探討如何利用 Music Transformer 模型進行 MIDI 音樂的資料前處理、模型訓練以及旋律生成，並建立一套能自動產生旋律片段的音樂生成系統，以協助創作者在旋律啟發、音樂草稿或創作輔助等情境中使用。

第二章 專題內容與進行方法

2.1 研究動機

音樂創作往往需仰賴靈感，而創作者在遇到瓶頸時常需反覆嘗試不同旋律、節奏與和聲組合。若能利用深度學習模型自動生成具音樂性的短旋律，將有助於提供靈感素材，減少創作初期的摸索時間。此外，相較於語言模型的資料形式，音樂資料具有更複雜的時間與表現維度，因此如何正確編碼、表示與訓練音樂序列，是一項值得研究與實作的主題。本專題希望透過 Music Transformer 探索音樂生成的可能性，並驗證其在 MIDI 事件序列上的生成能力。

2.2 研究目的

本專題旨在完成以下目標：

1. 建立完整的 MIDI 音樂資料前處理流程，包含事件序列轉換(encoding)、音符事件處理與資料集格式統一化。
2. 利用公開 MIDI 資料訓練 Music Transformer 模型，並探索合適的事件類型（如 Note-On、Note-Off、Time-Shift、Velocity）。
3. 開發音樂生成流程，能夠輸入初始種子事件，並自動生成具旋律延展性的音樂片段。
4. 建置可視化與音樂回放環境，讓使用者能直接聆聽模型生成的結果。
驗證 Music Transformer 在事件序列音樂生成上的可行性與實用性。

2.3 系統架構概述

本專題系統主要由三大部分組成：

1. 資料前處理模組：

將網路公開的 MIDI 檔案轉換成模型可處理的事件序列格式，例如 Note-On、Note-Off、Velocity、Time-Shift 等事件類型，並將其編碼成整數序列。

2. Music Transformer 訓練模組：

採用相對位置編碼與多頭自注意力機制，透過事件序列學習音樂的時間結構、節奏模式與旋律關係。

3. 音樂生成模組：

透過訓練後的模型，使用指定的 seed events 產生後續事件序列，最後轉回 MIDI 供使用者播放。

2.4 資料來源與資料處理方法

2.4.1 資料來源

本專題使用來自網路公開的 MIDI 音樂資料集，包括個人網站、開源資料庫及開放分享之 MIDI 資料。由於 MIDI 格式能同時保留音高、力度與時間資訊，適合作為事件序列模型的訓練素材。

2.4.2 事件序列(Event-based) 編碼方式

為符合 Music Transformer 的輸入需求，本專題採用 Magenta 常用的事件序列格式，包括：

- Note-On 事件：音符開始（128 種可能）
- Note-Off 事件：音符結束
- Time-Shift 事件：時間往後移動（以最短時間單位切分）
- Velocity 事件：音符力度類別化後的表示

MIDI 讀取後，會依序轉換為上述事件並編碼成整數，形成最終模型的訓練序列。

2.5 Music Transformer 模型架構

本專題採用的 Music Transformer 包含以下特點：

1. **相對位置編碼(Relative Positional Encoding)**：
使模型能學習音符在時間軸上的相對關係，提升音樂長距依賴性。
 2. **多頭自注意力(Multi-head Self-Attention)**：
讓模型能同時從不同角度觀察旋律，理解節奏、和聲與音高變化。
 3. **事件序列輸入方式**：
使模型更能直接理解 MIDI 的結構，而非傳統的固定時間切片法。
模型最後以自回歸方式進行序列生成。
-

2.6 模型訓練流程

訓練流程如下：

1. **資料讀取**：載入所有網路公開 MIDI 資料。
2. **事件轉換**：將 MIDI 轉換為事件序列。
3. **序列切割**：將事件序列切分為固定長度的訓練樣本。
4. **模型訓練**：設定學習率、批次大小、序列長度等超參數，並透過 GPU 執行訓練。
5. **Loss 監控**：以交叉熵作為損失函數，逐步調整參數。
6. **模型保存**：訓練完成後儲存 checkpoint 以供後續生成。

2.7 音樂生成方法

生成階段包含以下步驟：

1. 初始種子事件 (Seed Events)

可為隨機事件或由 MIDI 開頭截取。

2. 自回歸生成後續事件

模型依序輸出下一個事件，並將其加入序列中再次送回模型。

3. 事件序列轉回 MIDI

將生成的事件列表重新組合為具有音高、時間與力度的 MIDI 檔案。

4. 提供視覺化與播放介面

讓使用者能聽取生成片段並觀察鋼琴卷軸 (Piano Roll)。

2.8 系統開發工具與環境

- Python 3.x
- TensorFlow / PyTorch (依你 notebook 選用的版本)
- Magenta / NoteSequence 套件
- Google Colab (GPU 加速訓練)
- MIDI 解析：mido、pretty_midi

第三章 專題流程與架構

本章將說明本專題系統的整體架構、各模組的功能與流程，並以流程圖方式呈現資料由輸入至音樂生成的完整路徑。整體架構分為四大部分：資料取得與前處理、事件序列轉換、Music Transformer 訓練、音樂生成與回饋介面。

3.1 系統整體架構

本專題旨在建立一套可自動生成 MIDI 旋律片段的音樂生成系統，因此架構設計採用模組化方式，使各階段能獨立運作，並保留擴充性。整體架構如同一條資料處理管線（Pipeline），從原始 MIDI 收集、資料編碼，到模型訓練與最終音樂生成均具備完整流程。

系統架構主要包含以下模組：

1. 資料來源模組：

從網路公開 MIDI 資料集中下載或匯入 MIDI 檔案。

2. MIDI 前處理模組：

對 MIDI 進行解析、清洗、統一化與節奏量化。

3. 事件序列編碼模組：

將 MIDI 轉換成 Music Transformer 所需的事件格式，例如 Note-On、Note-Off、Time-Shift、Velocity。

4. Model Training 模組（Music Transformer）：

利用事件序列進行模型訓練，包括資料切割、批次建構、損失監控與參數更新。

5. 音樂生成模組：

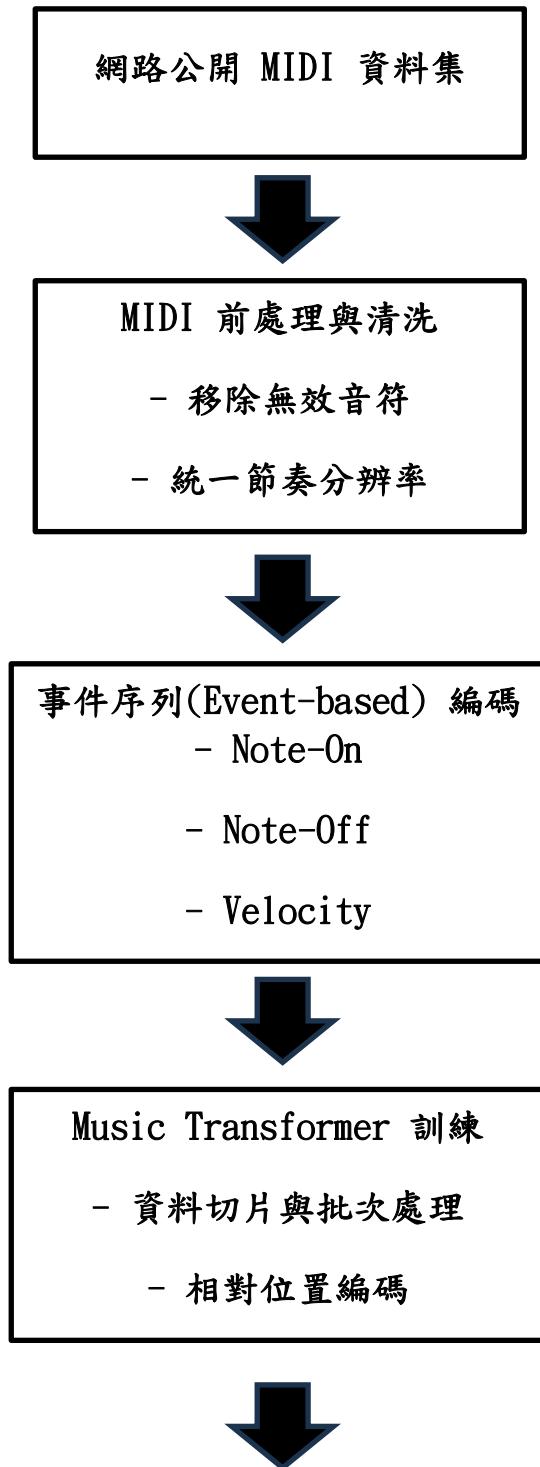
模型透過 seed events 生成後續事件，再轉換回 MIDI。

6. 播放與視覺化介面：

讓使用者能播放合成旋律，並搭配鋼琴卷軸或事件序列圖進行檢視。

3.2 系統流程圖

以下為本專題的主要資料處理流程：





音樂生成(自回歸)

- Seed Events 初始化

- 逐事件生成



MIDI 播放與可視化介面

3.3 各模組功能與設計理念

3.3.1 資料來源模組

本專題使用來源於網路公開 MIDI 的資料集，取材多元，包含鋼琴曲、流行音樂與旋律線明確的樂曲。為確保模型訓練品質，在匯入階段會排除：

- 長度過短的 MIDI
 - 無旋律或全以打擊樂構成之 MIDI
 - 資訊缺失或解析錯誤的檔案
-

3.3.2 MIDI 前處理模組

由於不同 MIDI 檔案的音軌、節奏與事件格式可能差異甚大，因此需在前處理階段進行統一化。主要處理項目包括：

- 解析多軌和弦與旋律資訊
- 統一 tempo 與 ticks per beat
- 刪除超出 pitch 範圍的音符
- 節奏量化與時間校正

前處理可確保輸入事件的一致性，提升模型可學習性。

3.3.3 事件序列編碼模組

Music Transformer 使用事件序列格式，因此需將 MIDI 音符轉成事件，例如：

- Note-On(n)：音符開始
- Note-Off(n)：音符結束
- Velocity(v)：音符力度
- Time-Shift(t)：時間往後移動 t 單位

此種格式能直接描述音樂的播放邏輯，模型因此能更有效捕捉音樂的時間依賴性與旋律關係。

3.3.4 Music Transformer 訓練模組

本專題採用 Magenta 提出的相對位置編碼(Relative Positional Encoding)，能改善長序列生成時的記憶能力，使模型更能掌握音樂中跨小節、跨樂句的長距關係。

訓練流程包括：

- 序列固定長度切片
- 建構訓練批次
- 以自回歸方式預測下一事件
- 損失函數為 cross entropy
- 使用 GPU 加速訓練

模型學習事件間的邏輯順序，並隨訓練逐步生成更具音樂性的旋律。

3.3.5 音樂生成模組

生成階段流程如下：

1. 使用 seed events 作為序列起始點
2. 模型預測下一事件
3. 將新事件加入序列並再次輸入模型
4. 重複直到達到指定長度
5. 將事件序列轉回 MIDI

生成後的 MIDI 可立即在介面播放，供使用者評估或進行創作參考。

3.4 系統優點

本專題架構具以下優勢：

- 模組化架構，易於維護與擴充
 - 以事件序列方式表示音樂，生成表現更自然
 - Transformer 架構具長距離依賴建模能力
 - 可處理任意 MIDI 來源，不侷限特定資料集
 - 生成結果可直接於介面播放並視覺化
-

3.5 本章小結

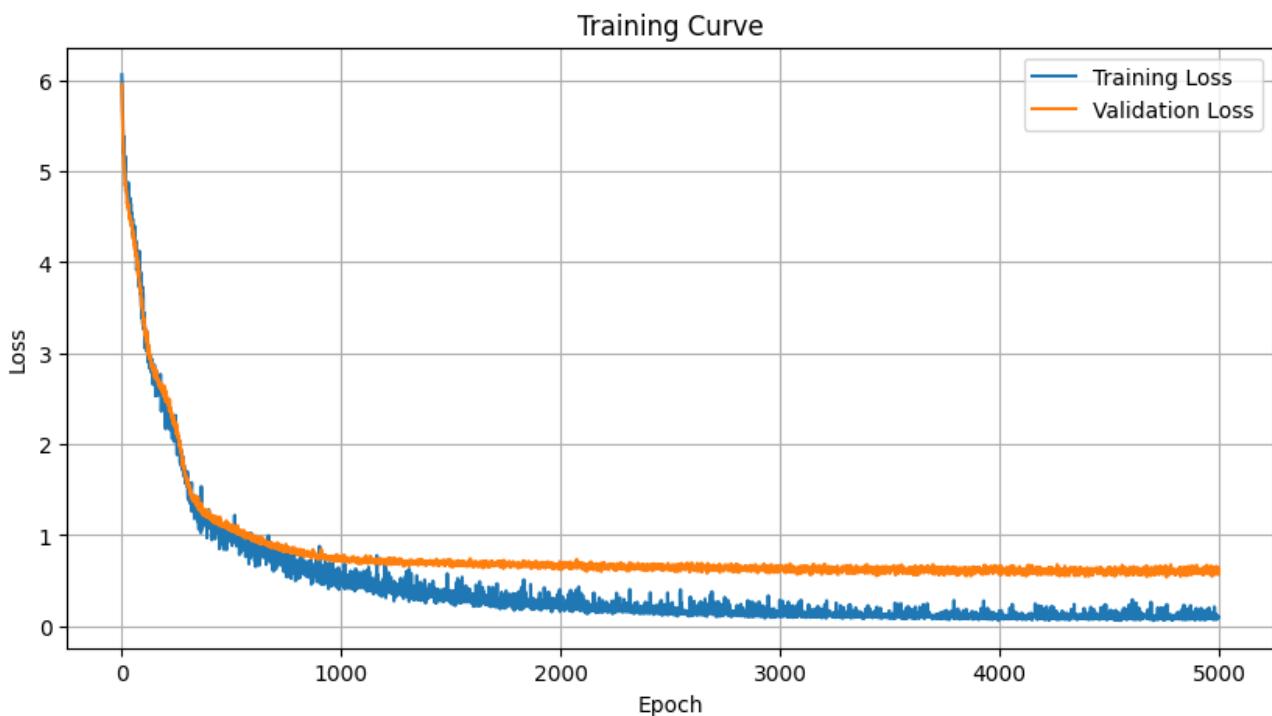
本章說明了本專題的整體架構、主要模組功能與資料處理流程。透過明確的模組設計與事件序列化方法，本專題得以順利建立 Music Transformer 訓練與音樂生成管線，為後續的系統實作奠定基礎。

第四章 專題成果介紹

本章將呈現本專題在 Music Transformer 模型訓練、樂曲生成、以及系統操作介面等多項成果。透過訓練曲線（Training Curve）與 MIDI 生成結果，我們得以評估模型的學習品質與音樂生成能力。

4.1 模型訓練成果

本專題使用整理後的網路公開 MIDI 資料訓練 Music Transformer，並將 MIDI 事件進行 Token 化，以利模型學習音符的啟動（Note-On）、結束（Note-Off）與持續時間（Duration）等音樂行為。為評估模型訓練狀況，我們記錄 Training Loss 與 Validation Loss 隨 Epoch 變化的情形，其結果如圖 1 所示。



(圖 4-1：模型訓練曲線)

圖 4-1 Music Transformer 模型訓練過程之 Loss 收斂曲線說明
本圖呈現模型在 5000 個 Epoch 中 Training Loss 與 Validation Loss 的變化軌跡。可以觀察到，在訓練初期 Loss 值迅速下降，代表模型快速掌握音符事件的基本關聯性，例如節奏模式與音符分布。隨著訓練持續進行，曲線逐漸趨緩，但仍保持穩定下降，顯示模型正在持續學習更高層次的音樂結構，如旋律走向與段落組織等。

此外，Validation Loss 與 Training Loss 之間的差距不大，表示模型並未發生明顯的過度擬合 (Overfitting) 情形，具備良好的泛化能力。此訓練結果顯示 Music Transformer 已成功收斂，具有穩定且可靠的音樂生成品質。

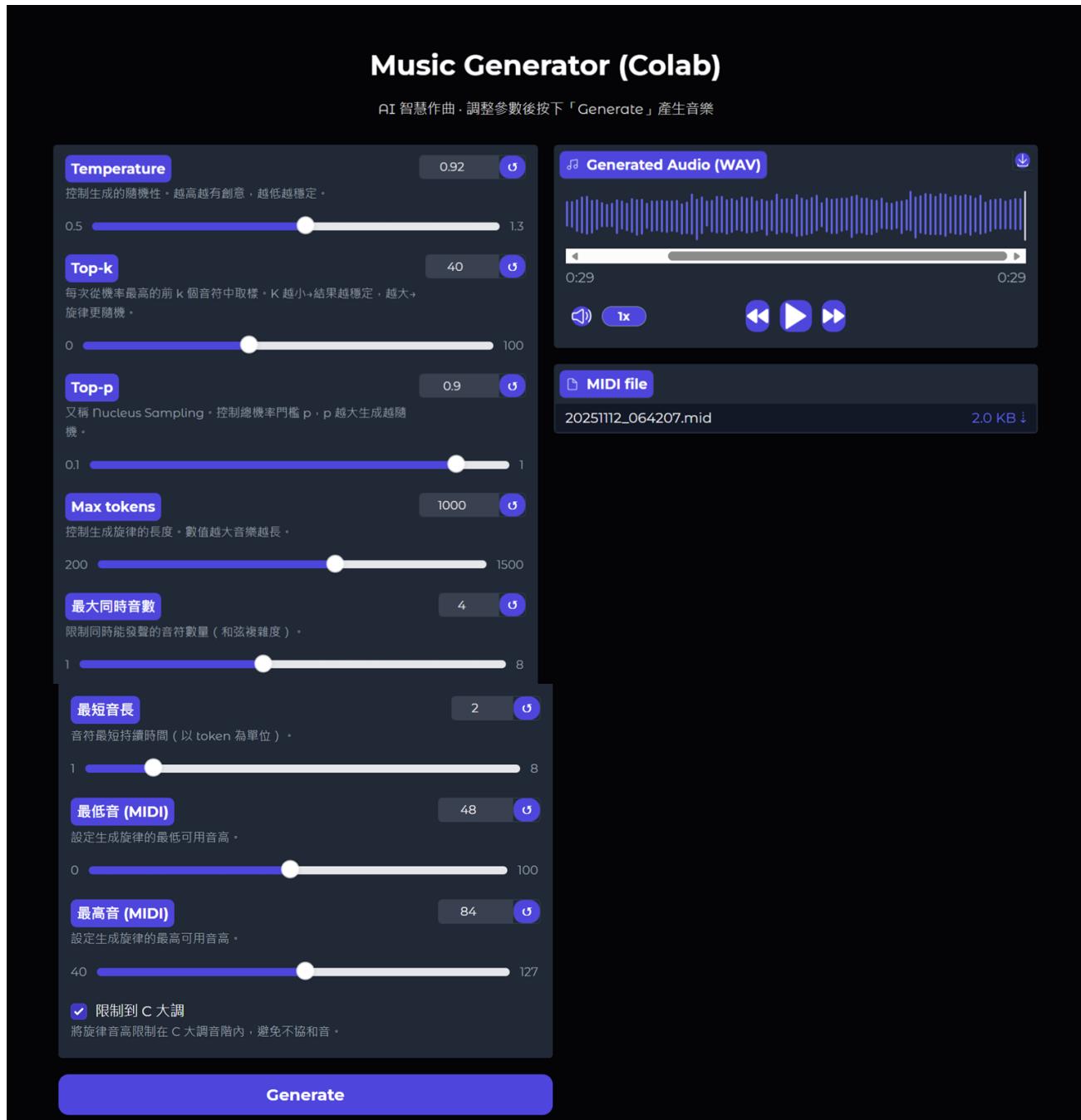
4.2 生成樂曲成果

完成模型訓練後，我們以訓練完成的 Music Transformer 模型產生旋律序列，並將生成的 Token 轉換回 MIDI 格式。生成結果包含旋律片段、節奏變化、音程走向等音樂行為。在大多數情況下，模型能夠產出具邏輯性的旋律句法，並呈現一定程度的風格一致性與音樂流暢度。

使用者可透過播放生成的 MIDI 片段，檢視模型所產生的樂曲特性，並進一步評估旋律的創意性與可聽性。整體而言，生成的旋律具備基礎音樂結構，展現 Music Transformer 對 MIDI 資料中特徵的有效學習。

4.3 系統介面與操作成果

本專題整合了簡易的操作介面，使使用者能直覺輸入參數並生成樂曲。介面強調功能清晰、流程簡單，讓沒有程式背景的使用者也能輕易體驗 AI 音樂生成。



(圖 4-2 系統主畫面及生成設定與輸出畫面)

系統主畫面提供參數調整、生成按鈕、以及 MIDI 檔案輸出等功能，呈現完整的音樂生成功能流程。

此區域讓使用者可：

- 輸入起始 Token 或選擇隨機初始化
- 設定旋律生成長度
- 產生 MIDI 音樂並進行播放或下載
- 查看生成狀態與基本模型資訊

透過此介面，使用者能快速體驗 AI 音樂生成的操作流程，並觀察不同參數對生成結果的影響。

4.4 本章小結

本章總結 Music Transformer 的訓練成果、生成音樂表現以及系統操作界面展示。透過 Loss 曲線可確認模型成功收斂；在旋律生成部分，模型能產生具邏輯性的音樂片段，呈現一定的音樂結構與流暢度。系統介面的整合則讓整個操作流程更直覺、易用，達成本專題預期之核心目標。

第五章 專題學習歷程介紹

本章將回顧本專題自發想、學習相關軟體工具，到實際系統實作過程中所經歷的學習歷程與問題解決方式。相較於前幾章偏重於系統架構與成果展示，本章著重於「學習過程」本身，包括：對開發工具與深度學習框架的熟悉、對 MIDI 與事件序列表示方式的理解，以及在模型訓練與音樂生成過程中所遭遇困難與因應策略。

5.1 專題相關軟體學習介紹

在本專題的實作過程中，團隊需同時面對「音樂領域知識」與「深度學習實作」兩個面向，因此在軟體與工具的學習上也較為多元。本節將依不同工具與平台進行說明。

5.1.1 Python 與 Google Colab 之應用

本專題主要以 Python 作為開發語言，並使用 Google Colab 作為主要執行環境。

學習重點包含：

1. 虛擬環境與套件管理：

在 Colab 中安裝並管理 torch、miditoolkit、mido、pretty_midi、tqdm 等套件，理解如何使用 pip 安裝與版本相容性問題。

2. GPU 資源使用：

在 Colab 啟用 GPU 加速，理解 GPU 與 CPU 訓練速度差異，以及因 GPU 記憶體限制而需要調整批次大小 (batch size)、模型維度與序列長度 (MAX_LEN) 等超參數的取捨。

3. 程式模組化與 Notebook 流程：

將程式區分為「資料前處理」、「事件編碼」、「模型定義」、「訓練流程」、「生成與後處理」等區塊，讓程式碼在 Notebook 中具備清楚的執行順序與重複利用性。

5.1.2 MIDI 處理與事件序列相關套件

由於本專題採用 MIDI 作為音樂資料的主要來源與輸出格式，因此 MIDI 解析與轉換是重要的一環。

在工具層面，主要學習內容如下：

1. MIDI 解析套件：

使用 miditoolkit / mido / pretty_midi 等套件讀取 MIDI 檔案，取得音符 (pitch)、起訖時間 (start / end)、音軌資訊與 velocity 等資料，並將多軌資料整理為適合模型學習的形式。

2. 事件序列 (Event-based Representation) 設計：

參考 Music Transformer 與 Magenta 常用格式，學習如何將 MIDI 轉為事件序列：

- Note-On(pitch)：音符開始
- Note-Off(pitch)：音符結束
- Time-Shift(Δt)：時間往後移動
- Velocity(v)：力度資訊

並在程式中自行定義 token 編號範圍與詞彙表大小 (VOCAB_SIZE)，搭配時間量化（例如以固定 ticks 作為最小單位）來控制序列長度與解析度。

3. 資料前處理與資料增強：

包括移除打擊樂軌道、過短音符、修正不合理音域，以及對 MIDI 進行移調 (Transpose)，增加訓練樣本多樣性，同時保持旋律結構。

5.1.3 深度學習框架與 Music Transformer 架構

在模型層面，本專題主要學習與應用以下概念：

1. 深度學習框架 (PyTorch / TensorFlow) :

實際在 Colab 上實作 事件序列 Transformer 模型，包含：

- Embedding：將離散 token 映射到向量空間
- 位置編碼或相對位置概念
- 多頭自注意力 (Multi-head Self-Attention)
- 前饋層 (Feed-forward layers)
- 最後以線性層輸出對下一個事件的機率分布

2. Music Transformer 概念理解：

透過文獻與 Magenta 教學資源理解 Music Transformer 的核心精神：

- 利用事件序列表示音樂，而非固定時間切片
- 使用相對位置編碼 (Relative Positional Encoding) 來處理長程依賴
- 以自回歸方式 (Autoregressive) 逐一產生新的事件，完成旋律片段生成

在實作上，雖然本專題採用的是「自行實作的小型 Transformer 架構」，但設計理念與 Music Transformer 一致，亦有助於實際理解論文中的設計思想。

5.1.4 音樂播放與視覺化工具

為了讓使用者直觀體驗模型輸出的結果，團隊也學習了：

1. MIDI 轉 WAV 音檔：

利用 Colab 上安裝之 FluidSynth 與 SoundFont，將生成的 MIDI 轉為 WAV 音檔，方便快速播放與比對不同參數設定下的聽感差異。

2. 簡易介面與互動式生成：

使用 Python 搭配 Gradio (或類似簡易介面工具) 建立一個網頁式操作介面，讓使用者可調整 temperature、top-k、top-p、音域、生成長度等參數，一鍵生成音樂並立即播放，提升系統的實用性與展示效果。

5.2 專題製作過程遭遇的問題與解決方法

在專題實作的過程中，團隊並非一路順利，而是經歷了多次失敗與修正。本節將依照實際開發流程，整理出較具代表性的問題與對應之解決方法。

5.2.1 資料與事件序列設計上的問題

(1) 不同 MIDI 檔案格式不一致

- 問題說明：

來源於網路的 MIDI 檔案常有多軌、鼓組、節奏標記不一致等情況，部分檔案甚至包含錯誤事件或極短、無法辨識的音符，若直接使用會影響模型學習。

- 解決方法：

- 在前處理階段撰寫清理程式：

- 移除打擊樂軌道
 - 刪除過短或零長度音符
 - 統一 ticks_per_beat 與 tempo 設定

- 建立基本檢查機制，若檔案格式異常則自動跳過，確保訓練資料品質。

(2) 事件序列長度與時間解析度的取捨

- 問題說明：

若時間解析度設太細，TIME_SHIFT 事件會非常多，導致序列過長、訓練成本偏高；若設太粗，又難以表達細膩節奏。

- 解決方法：

- 實驗不同 TS_RES (每個 TIME_SHIFT token 的 ticks 數)，平衡時間精細度與序列長度。
 - 設定 MAX_LEN 限制序列長度，並以滑動視窗或切片方式取固定長度片段作為訓練樣本。
 - 對 Velocity 做分級量化（例如 32 級），在保留力度變化的前提下控制詞彙表大小。

(3) 音域與不合理音高問題

- 問題說明：

部分 MIDI 可能存在極高或極低音符，導致生成時也學到不合常理的音域分布。

- 解決方法：

- 在前處理階段限制音域範圍，將超出範圍的音符捨棄或壓回合理區間。
 - 在生成階段加上「音域遮罩」，限制模型在解碼時只能選擇設定範圍內的音高。

5.2.2 模型訓練與超參數調整的問題

(1) 訓練不穩定與 GPU 記憶體不足

- 問題說明：

一開始若直接使用較大的模型維度與長序列，容易造成 GPU 記憶體不足，或訓練過程 Loss 震盪、學習不穩定。

- 解決方法：

- 採用「小型模型 + 梯度累積」方式：

- 減少單次 batch size，搭配 accum_steps 累積多次梯度再更新權重。

- 使用 OneCycleLR 等學習率排程策略，避免一開始學習率過高導致發散。

- 逐步調整模型參數（層數、維度、head 數），在可接受的訓練時間與 GPU 限制下取得平衡。

(2) 過度擬合與泛化能力不足

- 問題說明：

當訓練資料量有限時，模型容易記住特定旋律片段，導致生成結果過度接近訓練曲目，缺乏變化。

- 解決方法：

- 對 MIDI 進行移調資料增強，擴充實際訓練樣本數量。

- 分出驗證集並以 val_loss 監控模型泛化表現，於最佳點保存權重。

- 適度加入 Dropout、權重正規化等技巧，降低過度擬合風險。

5.2.3 生成階段的實務問題與修正

(1) 生成音樂過短或節奏破碎

- 問題說明：

即使設定較大的生成長度 (max tokens)，實際播出來的音樂仍可能只有短短幾秒，或是 TIME_SHIFT 太細碎導致聽感斷斷續續。

- 解決方法：

- 設定最小音長 (min_dur_tokens)，要求每個音至少持續一定時間。
- 在後處理階段加入節奏量化（例如將零碎的時間片段合併為固定拍格），讓節奏較為規則。
- 增加「總時長下限」，若累計時間不足則在序列尾端補上適當的 TIME_SHIFT。

(2) 過度重複的旋律與和聲

- 問題說明：

模型有時會反覆生成相同音型或和弦，導致音樂聽起來單調。

- 解決方法：

- 適度調高 temperature 或放寬 top-k / top-p，增加抽樣多樣性。
- 調整音域與調性限制，讓模型有更多可選擇的合法音高。
- 視情況加入簡單的「避免連續重複 token」規則，減少完全相同的事件長時間重複。

(3) 技術錯誤：MIDI 參數超出範圍與 CUDA 例外

- 問題說明：

實作過程中曾遇到：

- ValueError: data byte must be in range 0..127 (velocity 或 pitch 超出合法範圍)
- CUDA error: device-side assert triggered (生成階段遮罩與機率分布處理不當)

- 解決方法：

- 在事件轉回 MIDI 前，統一對 pitch 與 velocity 做「硬性夾範圍」，確保數值介於 0-127。
- 在解碼時先套用遮罩（禁止不合法或不符合規則的 token）再做 top-k / top-p 截斷，並在極端情況下設計 fallback 機制（例如至少允許最小 TIME_SHIFT），避免所有機率變為無窮小而導致 GPU 端錯誤。
- 若 GPU 狀態異常，則改以 CPU 測試程式邏輯，確認無邏輯錯誤後再重新啟動 GPU 訓練環境。

5.2.4 介面與使用體驗相關問題

(1) 參數調整不直觀

- 問題說明：

在僅有指令列的情況下，一般使用者較難理解 temperature、top-k、top-p 對音樂風格的影響。

- 解決方法：

- 以 Gradio 建立簡易網頁介面，提供滑桿與選項讓使用者直接調整參數，按下按鈕即可生成與播放結果。
- 於介面文字說明中，以「越保守／越隨機」、「旋律越穩定／越多變」等語句協助使用者理解參數意義。

(2) 生成結果的版本管理與比較困難

- 問題說明：

多次實驗不同參數或不同模型版本時，若未記錄生成條件，後續難以比較與重現。

- 解決方法：

- 在程式中自動為每次生成結果加上時間戳記與設定摘要（例如 temperature、top-k、top-p、音域），並寫入 JSON 或文字檔。
- 將生成的 MIDI/WAV 檔案與對應的參數記錄存放於同一資料夾，方便後續整理與撰寫報告。

5.3 本章小結

綜合而言，本章回顧了本專題在軟體工具學習與實作過程中所經歷的各項歷程。從最初對 Python、Colab、MIDI 處理套件與 Music Transformer 架構的摸索，到後續針對事件序列設計、模型訓練穩定度、音樂生成品質與介面操作體驗所做的調整與改進，皆反映出本專題不僅是「完成一個可以動的系統」，更是在過程中逐步理解：

1. 如何將抽象的音樂概念轉化為模型可學習的資料表示。
2. 如何在有限的計算資源與資料條件下，設計合理的模型與訓練策略。
3. 如何透過規則與後處理，讓生成式 AI 的輸出更符合人類對「音樂性」與「可聆聽性」的期待。

這些經驗對於未來進一步研究多軌和聲生成、風格控制、條件式音樂生成，以及整合音訊層面模型等方向，皆具有重要的參考價值。

第六章 結論與未來展望

6.1 研究結論

本專題以「AI 音樂生成」為主題，實作出一套以 MIDI 事件序列與 Transformer 架構為核心的音樂生成系統，並完成從資料前處理、事件編碼、模型訓練、旋律生成到介面呈現的完整流程。綜合前述各章內容，主要結論如下：

1. 事件序列表示法適合作為音樂模型輸入格式

透過將 MIDI 檔轉換為 Note-On、Note-Off、Time-Shift、Velocity 等事件序列，模型可以直接學習音符觸發時間、持續長度與力度變化等資訊，相較於傳統固定時間格點 (Time Grid) 或單純 Piano Roll 表示，事件序列在序列長度與表達彈性上更具優勢，有助於捕捉旋律、節奏與和聲的細緻變化。

2. Transformer 模型能有效學習長距離音樂結構

實驗結果顯示，Transformer 架構在處理較長的事件序列時，能維持較佳的樂句延續性與整體結構一致性，較不易出現完全隨機或中途崩壞的音符序列。相較於偏向短期依賴的傳統 RNN/LSTM，Transformer 在多小節旋律與段落層級結構上的表現較佳，更適合作為音樂生成的核心模型。

3. 資料前處理與資料品質對生成結果影響巨大

在前處理過程中，包含：

- 移除打擊樂與不合適音軌
- 修正零長度或過短音符
- 統一 tempo 與 ticks-per-beat
- 限制音域範圍與移調資料增強

這些步驟顯著提升了模型訓練穩定度與生成旋律的合理性。實務上也證實，良好的資料清理與整理，往往比單純堆疊模型深度更能提升生成品質。

4. 解碼階段的「規則約束」能大幅提升可聆聽性

本專題在生成階段加入多項控制條件，例如：

- 最大同時音數 (max polyphony)
- 最短音長 (min duration tokens)
- 音域限制 (pitch range)
- 調性限制 (如 C 大調)

搭配 top-k / top-p 抽樣以及適度的 temperature 設定，可以有效減少雜訊式輸出與極端不協調音，讓生成旋律更具音樂性與結構感。這顯示「深度學習模型 + 簡單樂理規則」的混合式設計，是現階段實作 AI 音樂生成的一條實用路線。

5. 系統已具備作為創作輔助工具的可行性

經由多次生成與主觀聆聽評估，本系統能夠產生：

- 節奏連貫且旋律可辨識的短樂句
- 在限制條件下具有一定風格傾向的片段

雖然尚未達到專業作曲水準，但已足以作為「創作草稿產生器」或「靈感輔助工具」，協助創作者在旋律發想到和聲實驗等情境中加速流程。

綜合以上，本專題成功驗證了基於事件序列與 Transformer 的 AI 音樂生成架構，在 MIDI 旋律生成的應用上具有實作可行性與實用潛力。

6.2 研究限制

儘管本專題已達成預期目標，但仍存在若干限制與不足之處，說明如下：

1. 資料規模與多樣性有限

本專題所使用的 MIDI 資料雖涵蓋部分風格，但整體資料量與風格多樣性仍有限，可能造成模型在特定節奏型態或常見音型上表現較好，但在少見風格或複雜和聲進行上仍顯不足。

2. 多軌與複雜和聲尚未完整處理

現階段系統較偏向單旋律或簡化和聲結構的生成，對於完整多軌編制（如左手伴奏 + 右手旋律、鋼琴 + 貝斯 + 鼓等）的支援較為初步，尚未建立完善的多聲部角色分工與編制控制機制。

3. 長篇結構與段落設計能力有限

雖然 Transformer 能處理較長事件序列，但在樂曲層級（如 AABA、Verse-Chorus-Bridge 等）的長程結構上，本系統尚未加入明確的段落控制與結構規劃，生成結果多以「片段旋律」為主，較少具備完整曲式輪廓。

4. 客觀評估機制仍有待加強

本專題在評估方面以損失函數（Loss）、Perplexity 及主觀聆聽為主，尚未導入更系統化的客觀指標，例如：

- 和聲合法性統計
- 音程跳進跳出比例
- 節奏多樣性與重複度分析

未來若能結合更多量化指標，將有助於更精準比較不同模型或不同參數配置的優劣。

6.3 未來展望

基於本專題所得之經驗與成果，未來可從以下幾個方向持續延伸與深化：

1. 多軌與自動配器 (Arrangement)

在現有單旋律生成基礎上，未來可發展多軌生成能力，例如：

- 自動生成左手伴奏與右手旋律
- 為既有旋律自動配置和聲或簡易伴奏型態
- 擴展至樂團編制（鋼琴+貝斯+鼓）之節奏組生成
此部分可結合角色標記（如「旋律軌」、「和聲軌」、「節奏軌」）與條件式生成模型。

2. 風格與條件控制 (Conditional Generation)

可嘗試在模型輸入中加入更多條件標記 (condition tokens)，例如：

- 樂曲風格 (Classical / Pop / Jazz)
- 調性與拍號 (Key / Time Signature)
- 節奏速度 (Tempo)
- 紿定簡單和弦走向或低音線

讓使用者能以「指定風格」或「指定情境」方式控制生成結果，提升實務應用價值。

3. 長篇結構與段落規劃

未來可考慮採用「兩階段生成」策略：

- 第一步：先生成節奏架構與段落配置（例如 8 小節主歌、8 小節副歌）
- 第二步：在固定結構中填入具體旋律與和聲
或引入更高層的「結構控制序列」，讓模型在生成過程中能意識到目前處於樂曲哪一段，提升長程一致性。

4. 與音訊生成模型結合

目前系統生成的是 MIDI 資料，音色與混音仍需仰賴外部 DAW 或虛擬樂器。未來可考慮：

- 將 MIDI 生成結果餵入音訊合成模型（如基於 Diffusion 或自注意力的音訊生成模型），直接產生具真實樂器音色的音檔。
- 探索「從旋律到編曲到音訊」一條龍式的生成架構。

5. 互動式創作介面與教育應用

現有的 Gradio 或網頁介面已能讓使用者簡單調整參數並聆聽結果，未來可進一步發展：

- 互動式「續寫」功能：使用者輸入前幾小節，由模型延續創作。
- 提供樂理視覺化輔助，作為音樂創作與教學輔助工具。
- 將本系統包裝為簡易教學平台，讓非資工背景的音樂系學生也能體驗 AI 作曲流程。

6.4 專題心得與自我成長

透過本專題的實作，組員在以下幾個層面有明顯成長：

1. 跨領域整合能力

需要同時理解程式設計、深度學習理論與音樂樂理，並將其整合到一套實際可運作的系統中。這種跨領域整合的過程，讓團隊更能體會理論與實務之間的距離與連結。

2. 問題分析與除錯能力

從資料處理錯誤、MIDI 參數超出範圍、GPU 記憶體限制，到生成階段的各種例外，過程中不斷需要閱讀錯誤訊息、查閱文件與重新設計程式流程。這些經驗都具體提升了成員面對未知問題時的耐心與解決能力。

3. 對 AI 與創作關係的再思考

在實際聆聽大量生成結果後，組員更加理解：

- AI 擅長的是「大規模樣式的學習與模仿」
- 真正的創作仍然需要人來下判斷、做取捨與賦予意義

AI 音樂生成因此較適合作為「輔助與啟發」的工具，而非取代創作者本身。

6.5 本章小結

本章綜合了本專題的實作成果，歸納出事件序列 + Transformer 於 MIDI 音樂生成上的可行性與優勢，同時也誠實面對資料、多軌、長程結構與評估方法等面向的限制。

展望未來，若能在多軌編制、風格控制、長篇結構規劃與音訊生成方面持續延伸，並搭配更友善的互動介面，本專題所建立的基礎架構，有機會發展為一套真正能在創作實務與教學現場中發揮功能的 AI 音樂創作輔助系統。

參考文獻

★ 期刊與論文 (Papers)

1. Huang, C. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., & Eck, D. (2018). *Music Transformer: Generating Music with Long-Term Structure*. *International Conference on Learning Representations (ICLR)*.
2. Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2018). *This Time with Feeling: Learning Expressive Musical Performance*. *arXiv:1808.03715*.
3. Payne, C. (2019). *MuseNet*: A Deep Neural Network That Generates Multi-Instrumental Music. *OpenAI Blog*.
4. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). *Attention Is All You Need*. *Advances in Neural Information Processing Systems (NeurIPS)*.
5. Dong, H. W., Hsiao, W. Y., Yang, L. C., & Yang, Y. H. (2018). *MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment*. *AAAI Conference on Artificial Intelligence*.

★ 技術工具與官方文件 (Frameworks & Libraries)

6. Google Magenta. (2020). *Magenta: Music and Art Generation with Machine Learning*. <https://magenta.tensorflow.org/>
7. Paszke, A., Gross, S., Massa, F., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. *NeurIPS*.
8. Mido MIDI Library. (2022). *Mido: MIDI Objects for Python*. <https://mido.readthedocs.io/>
9. PrettyMIDI. (2014). *PrettyMIDI: A Python Library for MIDI Processing*. <https://github.com/craffel/pretty-midi>
10. FluidSynth. (2020). *FluidSynth - Real-time Software Synthesizer*. <https://www.fluidsynth.org/>

★ 技術文件與參考資源 (Technical Resources)

11. Magenta Team. (2020). *Event-based Representation for Music Modeling*. Google Research.
12. TensorFlow Documentation. (2021). *Sequence Models and Transformers*. <https://www.tensorflow.org>

13. PyTorch Documentation. (2021). `nn.Transformer` API Reference.
<https://pytorch.org/docs>

★ 資料集

14. 奇幻音樂廳 https://mabinogi.fws.tw/ac_composer.php