和弦辨識競賽報告

一、 系統架構及方法說明

1. 系統硬體
   CPU：Intel i7-9700
   RAM：：Micron Ballistix Sport DDR4-3200 8G x2
   GPU：GIGABYTE GeForce RTX 2070 WINDFORCE 8G

2. 系統軟體
   我使用 Python 的 tensorflow.keras 模組開發
   Python 版本：3.8.5
   Tensorflow 版本：2.4.0
   Numpy 版本：1.19.2
   Cuda：cuda_11.0.2_451.48_win10
   cuDNN：cudnn-11.0-windows-x64-v8.0.4.30

3. 模型架構
   我嘗試了兩個架構：第一個是自製的，純粹的 4 層 ReLU + 1 層 Softmax 的分類器；第二個則是網路上的論文 [1] 中的雙向注意力模型。第一個自製的模型在甜蜜點的分數大約是 35～45 之間；第二個的分數則只在 25～35 之間，我推測有可能是因為在論文中，訓練資料是用不同參數的 librosa 所產生的，所以無法達到論文中宣稱的 70 分。
   所以，我最終上傳的答案是用我自製的簡單模型估算出來的。

二、細節與改動說明

1. 模型的訓練資料
   我沒有使用其他資料集，也沒有直接從 Youtube 上把音樂抓下來自行用 librosa 製作不同參數的訓練資料。

2. 模型的資料輸入輸出
   我撰寫的程式的模式是輸入 x 幀資料，運算後輸出 x 幀估算的和弦。 在程式中，雖然單純讀入資料沒問題，但是對於每一幀，我都將前後多幀資料融合進此幀合成一筆，資料的容量將會劇烈增加，32GB 的 RAM 無法負荷這麼大的資料量。舉例來說，若要訓練一個輸入 101 幀資料並輸出 101 幀估計和弦的模型，每筆訓練資料則會融合前後各 50 幀的資料，導致資料量暴增 101 倍。
   對於這個問題，我使用解決方法是，每次訓練（Epoch）只輸入以 Numpy 隨機取樣出的部分資料；並且，在模型的每次 訓練（Epoch）前，才對資料進行前後融合，然後每次在訓練（Epoch）後就將其拋棄，如此一來就可以解決 RAM 容量

不足的問題。但這也導致我的模型在學習時，不論是訓練資料或是驗證資料都與原本的資料集有一定的誤差，頂多只能在統計學的理論上達到高度相似，無法避免誤差的部分。

三、使用的外部資源與參考文獻

[1] A BI-DIRECTIONAL TRANSFORMER FOR MUSICAL CHORD RECOGNITION
https://arxiv.org/pdf/1907.02698v1.pdf

四、其他

1. Github
   雖然我從中途開始就沒做很詳盡的更新，但應該可以算是個開發證明吧。
   https://github.com/aisu-programming/Chord-Estimation