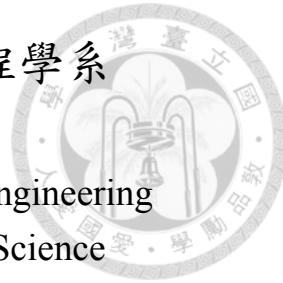


國立臺灣大學電機資訊學院資訊工程學系
碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis



改善基於神經網路與地標法的音訊指紋
Improvement of Neural Network- and Landmark-based
Audio Fingerprinting

陳羿豐

Yi-Feng Chen

指導教授：張智星博士
Advisor: Jyh-Shing Roger Jang, Ph.D.

中華民國 110 年 10 月
October, 2021

國立臺灣大學碩士學位論文
口試委員會審定書

改善基於神經網路與地標法的音訊指紋

Improvement of Neural Network- and Landmark-based
Audio Fingerprinting

本論文係陳羿豐君（學號 R08922077）在國立臺灣大學資訊工程
學系完成之碩士學位論文，於民國 110 年 10 月 24 日承下列考試
委員審查通過及口試及格，特此證明

口試委員：

張智生

王崇吉 (指導教授) 李子鴻

系主任

洪士顥

誌謝



這篇論文能夠成型，都要感謝我的指導教授張智星。感謝您提供我研究的環境、電腦設備，讓我能夠專心的做研究，也感謝您及時提供研究題目，我才有可能在兩年的時間畢業。

接下來要感謝實驗室的朋友們：王智穎、楊晨弘、劉濬慶、邱淳浩、卓書宇、簡義、陳品媛、李松融、王俞禮、李永霖、陳炫均、吳睿得、Brian、王鈞右、李學翰…… 等等，你們會在實驗室和我聊天，隨時接受我問問題，一起討論作業，還有跟我一起去吃午餐晚餐。有了你們，我的生活就充滿了快樂。

再來要感謝家人，總是會一直關心我的身體狀況、研究進度、課業壓力。你們會在放假的時間請我吃大餐、放鬆心情，平日則會提醒我要去吃飯了、穿多一點，還會聽我訴苦，雖然我覺得一直接電話好煩，但你們讓我有信心可以持續完成學業。

感謝冠傑學長，在疫情期間還繼續留在實驗室，提供我論文以及口試排練的建議，還帶我到處出去吃晚餐。

感謝張財銘學長，你提供給我研究所推甄的資訊，還幫助我整理審查資料，幫助我找到張智星教授，讓我能夠考上研究所。

感謝學園團契的朋友，在論文寫不出來的時候，會祝福我順利寫完，即使是我情緒化的表現，還是能夠包容我，也許這就是上帝給的恩典吧。

感謝宿舍輔導員，你都會在研一舍聽我抱怨，還會默默地解決宿舍的問題，帶給我有趣的活動。

感謝國研院國網中心提供計算與儲存資源，協助本研究順利進行。

還有好多人寫不下了，那就感謝上帝吧。

摘要



音訊指紋是一種音樂檢索方式，可用來快速的從錄音中辨識出相符的音樂，其作法是從錄音檔抽取顯著的特徵，並將此特徵和資料庫中的音樂特徵做比對。由於錄音經常會受到雜訊干擾，因此音訊指紋需要有抵抗環境噪音的能力。過去音訊指紋的做法主要是傳統演算法，如 Avery Wang 提出的地標法，近年來基於深度學習的音訊指紋做法已逐漸成為主流，如 Google 提出的 Now Playing。此篇研究主要聚焦在 Sungkyun Chang 等人提出的神經網路法音訊指紋。本論文首先以 MIREX 音訊指紋資料集來評估神經網路法和地標法，顯示出神經網路法在以現實世界的錄音來測試時，精準度仍然不如傳統演算法。因此本論文提出了三種方法來改進神經網路法：二階段洗牌、資料擴增改良以及對查詢做多次時間位移，並在最後以支援向量機 (Support Vector Machine, SVM) 來整合地標法和神經網路法的結果。為了方便重現，實驗使用公開的 Free Music Archive 資料集，透過加入雜訊的方式生成查詢音檔，並依照雜訊的強度分別計算檢索精準度。實驗結果顯示本論文提出的改進方式能夠顯著的提升神經網路在強雜訊下的精準度，並使得神經網路法在現實世界錄音查詢的表現超越地標法。

關鍵字： 音樂檢索、音訊指紋、地標法、對比學習、二階段洗牌、資料擴增、支援向量機

Abstract



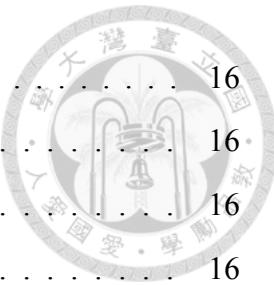
Audio fingerprint is a method in music information retrieval that can be used to quickly recognize matched music from an audio recording. To do that, it first extracts significant features from the recording file, and then compares these features with those extracted from database music. Since recordings are often contaminated with noise, audio fingerprint has to have the ability to resist background noise. In the past, the approaches to audio fingerprint were usually based on traditional algorithm, such as the landmark method, proposed by Avery Wang. Recently, audio fingerprint methods based on deep learning have gradually become mainstream, such as Google’s Now Playing. This work focuses on neural audio fingerprint, proposed by Sungkyun Chang et al. We first evaluated neural network method and landmark method on MIREX audio fingerprint dataset, and found that the accuracy of neural network method is still worse than traditional algorithm when tested with real-world recordings. Therefore we propose three approaches to improve such a method: two-phase shuffling, extensive data augmentation, and doing multiple time shifting to the query. Finally, Support Vector Machine (SVM) is used to integrate the results of the landmark method and neural network method. To make our work reproducible, we use public Free Music Archive dataset in our experiments and generate query audio by adding noise to this dataset. We then compute the query accuracy under different noise levels. Experiment shows that our approaches can significantly improve the accuracy of neural network under strong noise, and make neural network method perform better than the landmark method on real-world queries.

Keywords: music retrieval, audio fingerprinting, landmark method, contrastive learning, two-phase shuffling, data augmentation, SVM

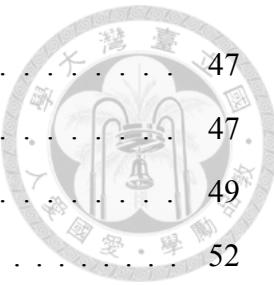
目錄



誌謝	ii
摘要	iii
Abstract	iv
第一章 緒論	1
1.1 主題簡介	1
1.2 研究方向與主要貢獻	2
1.3 章節概述	2
第二章 音訊指紋相關研究	4
2.1 地標法	4
2.1.1 時頻圖上建立顯著峰 (salient peaks)	4
2.1.2 將顯著峰轉為雜湊	5
2.1.3 比對並評分	6
2.2 Now Playing 法	7
2.2.1 Triplet loss	8
2.2.2 音樂偵測器	9
2.3 NAF 法	9
2.3.1 對比學習 (Contrastive Learning)	9
2.3.2 資料擴增 (Data Augmentation)	10
第三章 資料集簡介	12
3.1 MIREX 資料集	12
3.1.1 清理資料集	12
3.2 FMA 資料集	13
3.3 AudioSet	13
3.4 Aachen Impulse Response	14
3.5 Microphone Impulse Response Project (MicIRP)	14
3.6 資料集分割	14



第四章	研究方法	16
4.1	音訊指紋系統架構	16
4.2	地標法實作	16
4.2.1	資料前處理	16
4.2.2	抽取地標	17
4.2.3	將地標儲存到雜湊表	17
4.2.4	配對地標並計分	18
4.3	神經網路法實作	19
4.3.1	資料前處理	19
4.3.2	模型架構	20
4.3.3	資料擴增	20
4.3.4	損失函數 (loss function)	22
4.3.5	訓練方法	23
4.3.6	將內嵌向量儲存到資料庫	24
4.3.7	查詢與計分方式	26
4.4	評量指標	27
4.5	改進實驗	28
4.5.1	二階段洗牌 (Two-Phase Shuffling)	28
4.5.2	資料擴增改良	29
4.5.3	對查詢做多次時間位移	30
4.6	以 SVM 整合地標法和神經網路法的結果	31
4.7	實驗環境	32
第五章	實驗結果探討	33
5.1	基礎神經網路模型與地標法之比較	33
5.2	實驗一：二階段洗牌	34
5.3	實驗二：改良資料擴增-隨機時間位移	37
5.4	實驗三：改良資料擴增-雜訊強度	39
5.5	實驗四：改良資料擴增-殘響	41
5.6	實驗五：對查詢做多次時間位移	43

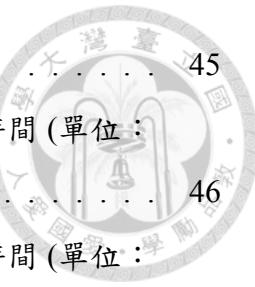


5.7 實驗六：以 SVM 整合地標法和神經網路法的結果	47
5.7.1 線性 SVM	47
5.7.2 RBF 核 SVM	49
第六章 結論與未來展望	52
6.1 結論	52
6.2 未來展望	54
參考文獻	56
附錄 A — 神經網路法效能分析	59



圖目錄

2.1 時頻譜上建立顯著峰	4
2.2 將顯著峰配對成地標	5
2.3 配對成功的時間特徵分布圖和時間差的直方圖	6
2.4 配對失敗的時間特徵分布圖和時間差的直方圖	7
2.5 Now Playing 模型架構	9
2.6 Cutout 和 SpecAugment	11
4.1 音訊指紋系統架構圖	16
4.2 地標轉成雜湊鍵和雜湊值的方式	18
4.3 神經網路的訓練與測試流程	19
4.4 空間可分卷積層	20
4.5 L_2 投射層	21
4.6 倒排索引	25
4.7 查詢與計分方式流程圖	27
4.8 二階段洗牌	29
4.9 查詢和原曲相差 0.25 秒的情況，片段內的數字表示對應到原曲的時間	30
4.10 對查詢做多次時間位移	31
5.1 基礎模型與地標法的精準度比較圖	33
5.2 二階段洗牌實驗在 FMA 驗證集 1 上的精準度	35
5.3 二階段洗牌實驗在 MIREX 資料集上的精準度	35
5.4 隨機時間位移實驗在 FMA 驗證集 1 上的精準度	37
5.5 隨機時間位移實驗在 MIREX 資料集上的精準度	38
5.6 雜訊強度實驗在 FMA 驗證集 1 上的精準度	40
5.7 雜訊強度實驗在 MIREX 資料集上的精準度	40
5.8 殘響實驗在 FMA 驗證集 1 上的精準度	41
5.9 殘響實驗在 MIREX 上的精準度	42
5.10 對查詢做多次時間位移實驗在 FMA 驗證集 1 上的精準度	44



5.11 對查詢做多次時間位移實驗在 MIREX 資料集上的精準度	45
5.12 對查詢做多次時間位移實驗在 MIREX 資料集上的檢索時間 (單位：秒)	46
5.13 對查詢做多次時間位移實驗在 FMA 驗證集 1 上的檢索時間 (單位：秒)	46
5.14 使用線性 SVM 的查詢精準度和 C 參數的關係，只考慮恰有一種方法正確的查詢	48
5.15 FMA 資料集 SNR 為 -10~10dB 的分數分布以及線性 SVM 的分界線	49
5.16 MIREX 資料集的分數分布以及線性 SVM 的分界線	49
5.17 使用 RBF 核 SVM 在 FMA 資料集 SNR 為 -10~10dB 的整體查詢精準度	50
5.18 使用 RBF 核 SVM 在 MIREX 資料集的整體查詢精準度	50
5.19 FMA 資料集 SNR 為 -10~10dB 的分數分布以及 RBF 核 SVM 的分界線	51
5.20 MIREX 資料集的分數分布以及 RBF 核 SVM 的分界線	51



表目錄

3.1 MIREX 音訊指紋資料集	12
3.2 FMA 資料集	13
4.1 模型架構	21
4.2 訓練機器規格	32
4.3 測試機器規格	32
5.1 基礎模型與地標法的精準度 (單位 : %)	33
5.2 基礎模型與地標法的精準度 (續) (單位 : %)	34
5.3 神經網路法與地標法的資料庫大小	34
5.4 二階段洗牌實驗精準度 (單位 : %)	36
5.5 二階段洗牌實驗精準度 (續) (單位 : %)	36
5.6 改良時間位移資料擴增實驗精準度 (單位 : %)	37
5.7 改良時間位移資料擴增實驗精準度 (續) (單位 : %)	38
5.8 改良雜訊資料擴增實驗精準度 (單位 : %)	39
5.9 改良雜訊資料擴增實驗精準度 (續) (單位 : %)	39
5.10 改良殘響資料擴增實驗精準度 (單位 : %)	41
5.11 改良殘響資料擴增實驗精準度 (續) (單位 : %)	42
5.12 對查詢做多次時間位移實驗精準度 (單位 : %)	43
5.13 對查詢做多次時間位移實驗精準度 (續) (單位 : %)	44
5.14 對查詢做多次時間位移的平均檢索時間 (單位 : 秒)	45
5.15 FMA 資料集驗證集 2 上，恰有一種方法正確的查詢個數	47
5.16 使用線性 SVM 的整體查詢精準度和 C 參數的關係 (單位 : %)	48
6.1 以 FMA 驗證集 1 評估各種改良實驗的精準度 (單位 : %)	53
6.2 以 FMA 驗證集 2 和 MIREX 資料集評估各種改良實驗的精準度 (單 位 : %)	53
A.3 神經網路法在搜尋時各個步驟的平均執行時間 (單位 : 毫秒)	59



A.4 神經網路法以 GPU 加速 KNN 搜尋，在搜尋時各個步驟的平均花費
時間 (單位：毫秒)

第一章 緒論



1.1 主題簡介

日常生活中我們到處可以聽到各種音樂，街道上、餐廳裡、影片配樂、或是身邊的朋友放音樂，可能會想再聽一次，但是常常不知道歌名、歌手，或者聽不出歌詞，因此無法找到音樂。在有歌詞的情況下可以用一般的搜尋引擎尋找歌詞，但是外文歌、純音樂、或是聽錯歌詞的話就難以用歌詞檢索。這時可以用音訊指紋來辨識出當下所聽到的歌曲。音訊指紋是一種適用於音樂的檢索系統 (retrieval system)，可從音訊中抽取顯著的特徵，以快速地從查詢音訊中找出相同的音訊。目前音訊指紋已有多個商業化的應用程式及服務，比如 Shazam [21]、SoundHound [22] 和 Google 助理¹等，它們提供手機 app，可讓使用者隨時錄製音樂片段，並透過網路傳輸音訊或指紋，由後端伺服器計算後回傳查詢到的歌曲給使用者。

音訊指紋之所以稱為音訊指紋，是因為在檢索時，將每個音訊視為獨特的指紋，即使混入環境噪音中，仍然可以辨識出其指紋。此技術因此可以精準的辨識出原曲，而不會將翻唱、雜訊、或其他音樂誤判為原曲。由於音訊指紋的特性，使得音訊指紋也被運用在不同的地方，如版權偵測、權利金分配、重複音樂偵測、聲音對齊等。音訊指紋為了能夠從環境雜訊中辨識出音樂，需要有抗雜訊的能力，同時也由於音樂資料的不斷增加，因此如何快速的檢索以及減少資料庫容量是音訊指紋的重要議題。

目前已有許多種音訊指紋的做法以及研究，其中最早提出的方法有 Wang 提出的地標法 [23]，和 Haitsma [13] 提出的以 Bark frequency cepstrum coefficient 及位元錯誤率 (bit error rate) 為基礎的方法。近來也出現了基於深度學習的音訊指紋，如 Now Playing [12]、SAMAF (Sequence to Sequence Autoencoder Model for Audio Fingerprinting) [5] 和 NAF (Neural Audio Fingerprint for High Specific Audio Retrieval Based on Contrastive Learning) [6]，其中 Now Playing 和 NAF 使用卷積神經網路，SAMAF 使用了長短期記憶 (long short-term memory) 架構。

¹<https://support.google.com/assistant/answer/7554088?hl=zh-Hant&co=GENIE.Platform%3DAndroid>



1.2 研究方向與主要貢獻

此篇研究主要聚焦在 Sungkyun Chang 等人提出的神經網路法音訊指紋 [6]，依序以各種改進實驗來提升其精準度，接著嘗試與 Wang 提出的地標法的結果做結合，以得到更高的精準度。

本論文以公開的 Free Music Archive (FMA) 音樂資料集 [8] 作為訓練資料，測試資料則有 FMA 資料集和台大多媒體資訊檢索實驗室²內部的 MIREX 資料集兩種。

本篇論文之主要貢獻如下：

- 以不同訊噪比 (signal to noise ratio, SNR) 的查詢資料來評估神經網路法 [6] 和地標法 [23] 音訊指紋的表現，分析其優缺點。
- 提出改良方法一：對訓練資料進行二階段洗牌，先打亂音樂，再打亂音樂片段。
- 提出改良方法二：調整資料擴增的方法，設法以資料擴增更真實的模擬查詢環境的干擾。
- 提出改良方法三：基於前人對地標法的改良 [3]，對查詢做多次時間位移，將其修改以適用於神經網路法，使查詢能更精準的對齊音樂，提高辨識率。
- 提出改良方法四：以支援向量機 (support vector machine, SVM) 整合神經網路法與地標法之查詢結果。

1.3 章節概述

本篇論文分為六個章節：

- 第一章緒論，簡介音訊指紋的應用場景。
- 第二章節相關研究，介紹現有的音訊指紋作法。
- 第三章資料集，介紹所有實驗中用到的資料集。

²<http://mirlab.org>

- 第四章研究方法，說明音訊指紋系統的實作方式、模型架構、以及改進實驗。
- 第五章實驗結果，以圖表展示各項改進實驗的結果，並比較分析優劣。
- 第六章結論，為本研究做結論，並討論未來展望。



第二章 音訊指紋相關研究



2.1 地標法

地標法是由 Avery Wang 提出的方法 [23]。此方法大致可以分成三個步驟：時頻圖上建立顯著峰、將顯著峰轉為雜湊、比對並評分。此方法的優點是對雜訊有極強的抵抗能力，並可以快速的尋找數百萬首音樂。知名的 Shazam 音樂辨識服務 [21] 即是使用此方法檢索。

2.1.1 時頻圖上建立顯著峰 (salient peaks)

Wang 進行多種實驗，以找出可以抵抗錄音雜訊以及 GSM 編碼失真的特徵。該篇最終採用的是時頻譜上的局部最大值的位置，因為局部最大值的位置比較不受到雜訊的影響 [23]。然而時頻譜上的局部最大值太多，難以檢索，而且多數的局部最大值強度不足，在錄音遇到雜訊時即消失，不具有抵抗雜訊的能力，因此還會依據局部最大值的能量篩選出較可能留下的點，稱為顯著峰 (salient peak)，如圖2.1。

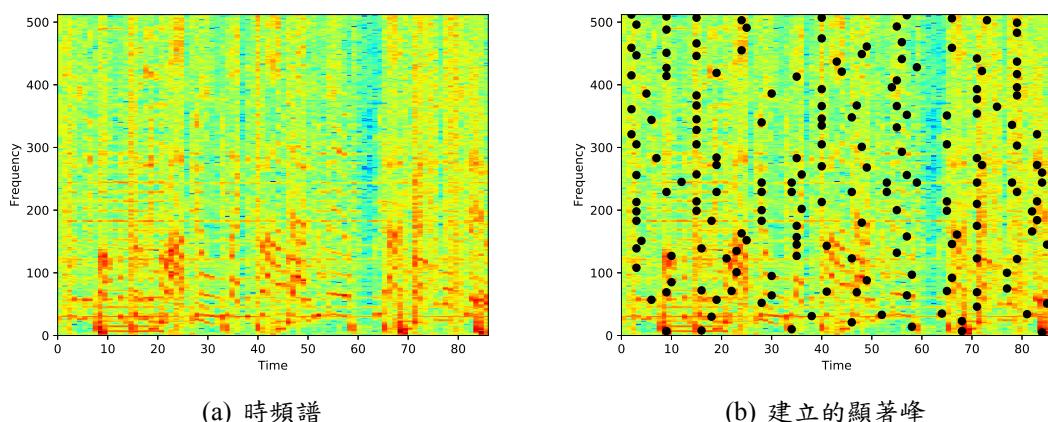


圖 2.1: 時頻譜上建立顯著峰 [23]



2.1.2 將顯著峰轉為雜湊

若是只以顯著峰做查詢，會非常的沒有效率。因此 Wang 的論文提出將顯著峰配對，以加速顯著峰的索引。配對的方法是，把每個顯著峰當作錨點 (anchor point)，在其右側畫出一個事先定義大小的矩形目標區域 (target zone)，目標區域內的顯著峰會和錨點配對，這些由兩個顯著峰組成的配對稱作地標 (landmark)。以圖2.2為例，黑點是顯著峰，紅點是其中一個錨點，其目標區域是黑色的矩形，藍點是和紅點配對的顯著峰，並以藍線表示構成的地標。

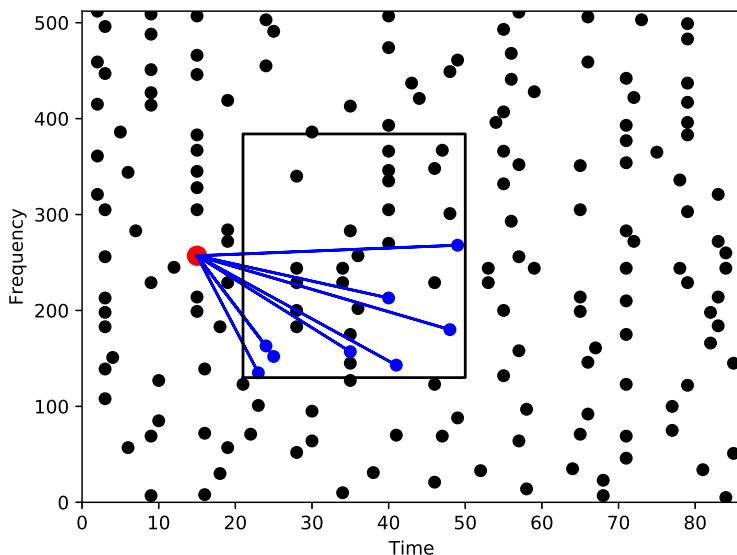


圖 2.2: 將顯著峰配對成地標

該論文為了減少資料庫的大小，限制一個錨點可以和其他顯著峰配對的個數，他們將此限制稱為扇出 (fan-out)，以 F 表示 [23]。假如 $F = 8$ ，顯著峰有 N 個，則最多可組成 $8N$ 個地標。限制 F 的數量，可以降低資料庫的大小，然而也可能因此降低配對的精準度。本論文在後續的實驗中，會把 F 的值設為 8。

接下來把地標轉成雜湊，以用來查詢。地標由兩個顯著峰 (t_1, f_1) 、 (t_2, f_2) 組成，其中 $t_1 < t_2$ ，則可以把地標轉成雜湊鍵 (hash key) $(f_1, \Delta f = f_2 - f_1, \Delta t = t_2 - t_1)$ ，由兩個顯著峰的頻率與顯著峰的時間差組成，雜湊值 (hash value) 是 (songID, t_1) 。如此只要輸入地標的雜湊鍵，就可以檢索所有相關的音樂以及其出現的時間點。



2.1.3 比對並評分

使用以上兩個步驟來建立音樂資料庫後，接下來就可以查詢音樂。查詢片段會經過上述的兩個步驟來取得顯著峰、地標和雜湊鍵，接著使用這些雜湊鍵從資料庫中檢索。如果查詢片段和某首音樂吻合，則查詢片段和音樂的地標應該有相似的相對位置。該論文 [23] 假設查詢的音樂只是原曲的時間位移，因此查詢片段和原曲的對應特徵的時間有以下的關係式：

$$t_{song} = t_{query} + \text{offset time} \quad (2.1)$$

其中 t_{song} 是原曲特徵的時間， t_{query} 則代表查詢片段對應到的特徵的時間，offset time 是查詢和原曲的時間差。因此若是某首音樂和查詢片段吻合，則 (t_{song}, t_{query}) 畫成分布圖的時候，會呈現出斜率為 1 的直線，如圖 2.3 (a)。若是不吻合，則分布圖不會呈現出斜率為 1 的直線，如圖 2.4 (a)。該論文提出一種快速找出時間差

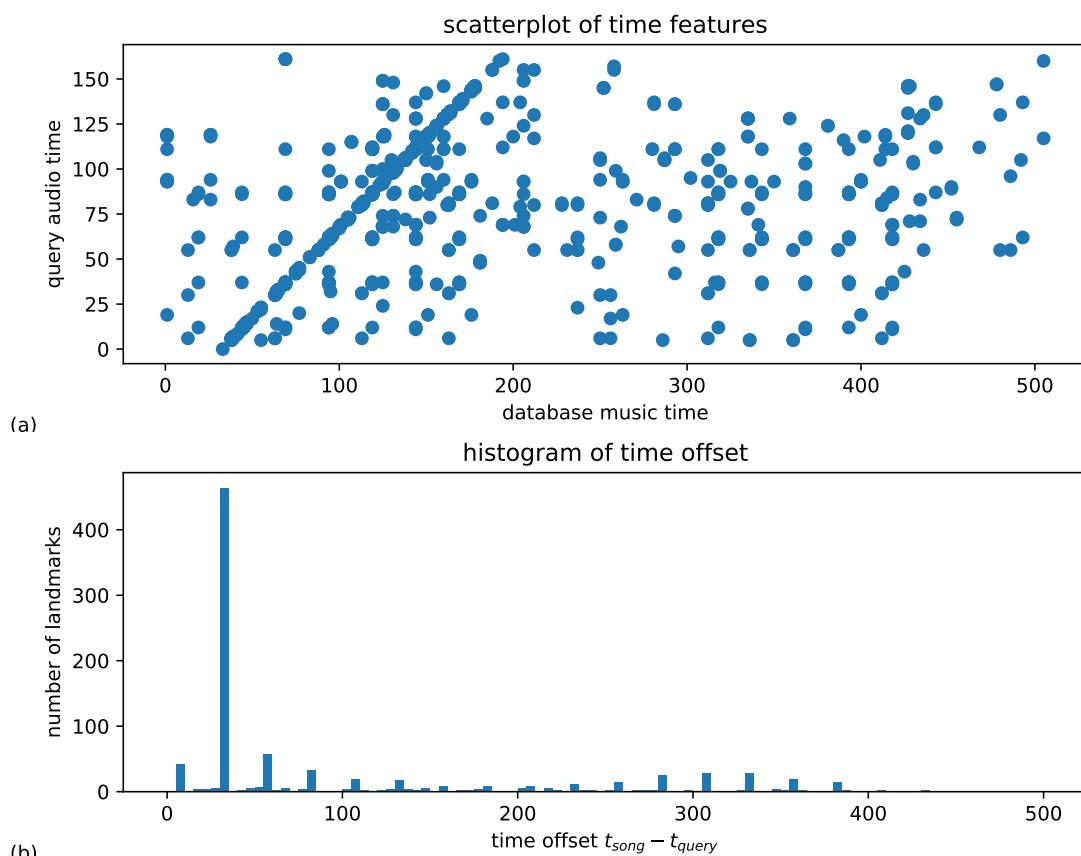


圖 2.3: 配對成功的時間特徵分布圖和時間差的直方圖 [23]



的方法，就是對於每個分布圖上的點 (t_{song}, t_{query}) ，計算

$$\delta t_k = t_{song} - t_{query}$$

則出現最多次的 δt_k 即為時間差，可以先將 δt_k 排序，再順序掃描 δt_k 以統計出現次數，時間複雜度是 $O(N \log N)$ ，N 代表分布圖的點個數。最後分數為出現最多次的 δt_k 的出現次數。觀察 δt_k 出現次數的直方圖，通常若某音樂確實出現在查詢裡，則會有某個 δt_k 出現的次數最高，如圖 2.3 (b)，反之若某音樂不存在於查詢中，則 δt_k 出現次數的直方圖較無明顯最高值，如圖 2.4 (b)。

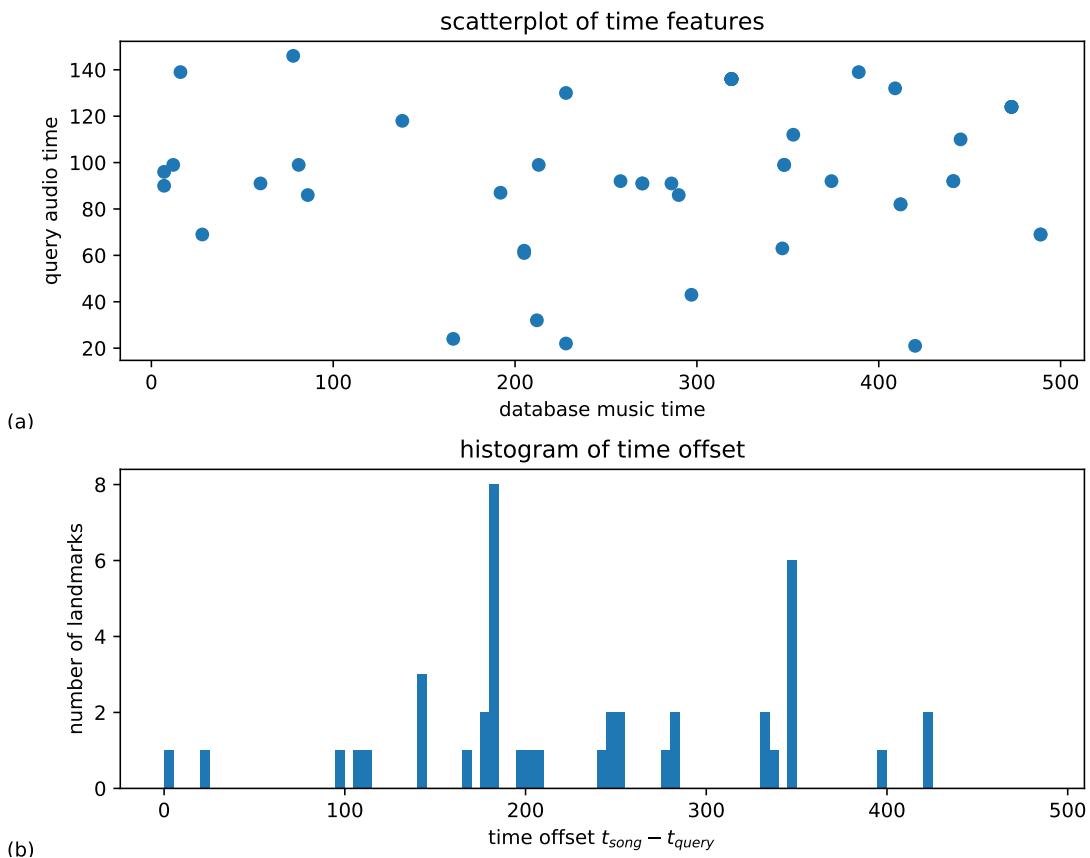


圖 2.4: 配對失敗的時間特徵分布圖和時間差的直方圖 [23]

2.2 Now Playing 法

此方法是 Google 在 2017 年提出 [12]，為使用類神經網路處理音訊指紋的開端。



Google 原本的目的是為了在行動裝置上能夠不斷地辨識周遭的音樂，而不需要伺服器運算。因此他們使用神經網路來抽取音訊指紋，使得比對音訊所需的資料庫大小能夠比傳統演算法更小。論文中宣稱每首歌平均只需要 3 KB 的資料庫空間 [12]。

該論文中神經網路的輸入是時頻譜，輸出是一個 96 維的向量，96 綴的向量會再量化成 96 個位元，作為音訊指紋的索引，大約每一秒的音訊會產生一個索引。神經網路的架構如圖 2.5，是先經由卷積層抽取內嵌向量 (embedding)，再把內嵌向量平分成 96 個子向量，各自經過 2 層全連通層，每個子向量會輸出 1 個值，最後把 96 個子向量產生的值連接成一個向量。

2.2.1 Triplet loss

由於資料庫的音樂至少有上萬首，而且會不斷增加，因此將音訊指紋當作一般分類問題來訓練是不可行的。Now Playing [12] 的神經網路輸出內嵌向量，再用內嵌向量做最鄰近搜索，因此不需要為每首音樂都設定一個分類，這種機器學習方式叫做度量學習 (metric learning)。為了讓神經網路可以輸出有效的內嵌，該論文使用 Triplet loss [20] 來訓練神經網路，Triplet loss 的訓練資料是三個訓練樣本 (x^a, x^p, x^n) 的配對，其中要求 x^p 是和 x^a 相同的樣本， x^n 是和 x^a 不相同的樣本。Triplet loss 的公式如下：

$$L = \sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+ \quad (2.3)$$

這個 loss 函數會促使模型把相同的樣本靠近，不相同的樣本之間則會有較大的距離， α 是和相似度距離有關的參數。在該論文 [12] 中，兩個樣本相同的定義是，從同一首音樂取出來的兩個片段，其開始時間相差不超過幾百個毫秒。原文並未說明確切的時間差範圍，本論文的設定詳見第 4.3.3 節的隨機時間位移部分，時間差的上限是 200 毫秒。



2.2.2 音樂偵測器

為了省電，該論文 [12] 還提出了用神經網路來偵測音樂是否存在，只有偵測到音樂存在時，才會啟動音訊指紋的神經網路。這使得整個音訊指紋系統一天只消耗不到 1% 的電力。

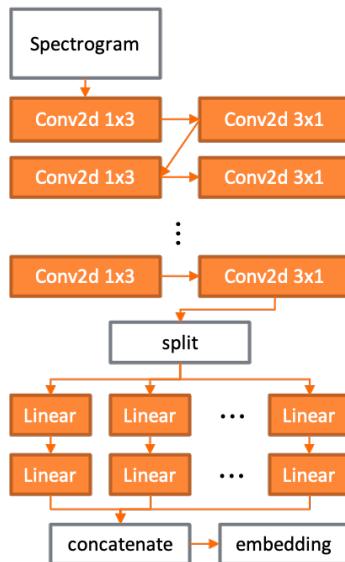


圖 2.5: Now Playing 模型架構 [12]

2.3 NAF 法

NAF 法是由 Chang 等人在 2021 年於 IEEE ICASSP [6] 發表，是基於 Now Playing 法的改良。由於 Now Playing 的論文並未公開資料集以及程式碼，因此該篇論文自行實作神經網路，並使用公開資料集來訓練。該論文採用和 Now Playing [12] 相同的音訊指紋模型基礎架構，但由於 Now Playing [12] 並未說明模型的層數以及輸入的格式，因此他們使用梅爾時頻譜作為神經網路的輸入，神經網路的層數則設定為 8 層卷積。

2.3.1 對比學習 (Contrastive Learning)

NAF 和 Now Playing 都使用對比學習，但使用不同的損失函數 (loss function)，NAF 論文 [6] 的損失函數是 Chen 等人 [7] 提出的 Normalized Temperature-scaled Cross Entropy Loss，簡稱 NT-Xent Loss。

NT-Xent Loss 的訓練資料是取出 N 個樣本，接著每個樣本都製造出兩個變種當作 mini-batch，這時 mini-batch 中的樣本兩兩配對，這些配對可以分成正樣本和負樣本，正樣本是來自同一個訓練樣本的兩個變種，負樣本則是來自不同訓練樣本的變種。在 NAF 論文中，每個樣本是音樂中取出的一個 1 秒片段，而產生的兩個變種中，其中一個是原始片段，另一個是由原始片段經由資料擴增產生的變種。

2.3.2 資料擴增 (Data Augmentation)

NAF 法 [6] 不使用真實的查詢片段來作為訓練資料，而是使用資料擴增的技術來模擬查詢時會遭遇的各種干擾，因此資料擴增顯得相當重要。他們使用的資料擴增有以下五種：

- 時間位移：由於查詢片段可以在原曲的任意時間錄製，但是神經網路的輸入為每 0.5 秒切出的音訊片段，因此即使是和查詢片段最相近的原曲片段也可能和查詢片段相差最多 0.25 秒。為了處理可能的時間差，訓練時，會從原曲先取出 1.2 秒的片段，再從這 1.2 秒的片段隨機取出兩個 1 秒鐘的片段，一個當作原始音樂，另一個當作受干擾的查詢。
- 混合背景雜訊：用來模擬查詢時的背景噪音。該論文使用原曲和雜訊之間的訊噪比 (signal to noise ratio, SNR) 來衡量雜訊的強度，SNR 的範圍設為 0 到 10dB。背景雜訊的資料集則包含 AudioSet [11]，和他們自行錄製的餐廳背景噪音。在接下來的論文中，SNR 都是指音樂訊號強度與雜訊強度的比值，SNR 越大，表示雜訊越弱，反之則表示雜訊越強。
- 殘響：用來模擬查詢地點的環境殘響和麥克風的頻率響應。做法是將音訊依序和環境殘響和麥克風的脈衝響應做卷積。環境殘響資料集是 Aachen [15] 資料集，麥克風資料集是 Public Microphone IR Project [24]。
- Cutout：由 DeVries 等人所提出的方法 [9]，原本是在影像上隨機的把一些矩形區域移除掉，作為一種簡單的 Regularization 的方法，在此處則是將時頻譜視為影像，隨機的選出一個矩形區域，並將該矩形區域填入 0，如圖 2.6 (a)。



在同一個 batch 裡的每個時頻譜都會選擇相同位置和大小的矩形。矩形區域的寬度和高度是時頻譜寬度和高度的 $1/10$ 到 $1/2$ 。

- SpecAugment：由 Park 等人提出的時頻譜上的 Data Augmentation [18]。SpecAugment 引入了頻率遮罩和時間遮罩，頻率遮罩會將一個頻率範圍內的所有時頻譜移除掉，時間遮罩則會將一個時間範圍內的時頻譜移除掉。經過遮罩後的時頻譜如圖 2.6 (b)。和 Cutout 一樣，在同一個 batch 裡的每個時頻譜都會選擇相同位置和大小的遮罩。時間和頻率遮罩的寬度各是時頻譜時間和頻率寬度的 $1/10$ 到 $1/2$ 。

在訓練階段，會使用到以上全部的資料擴增方法，但在測試階段產生查詢片段時，只會使用到混合背景雜訊和殘響兩種方法。

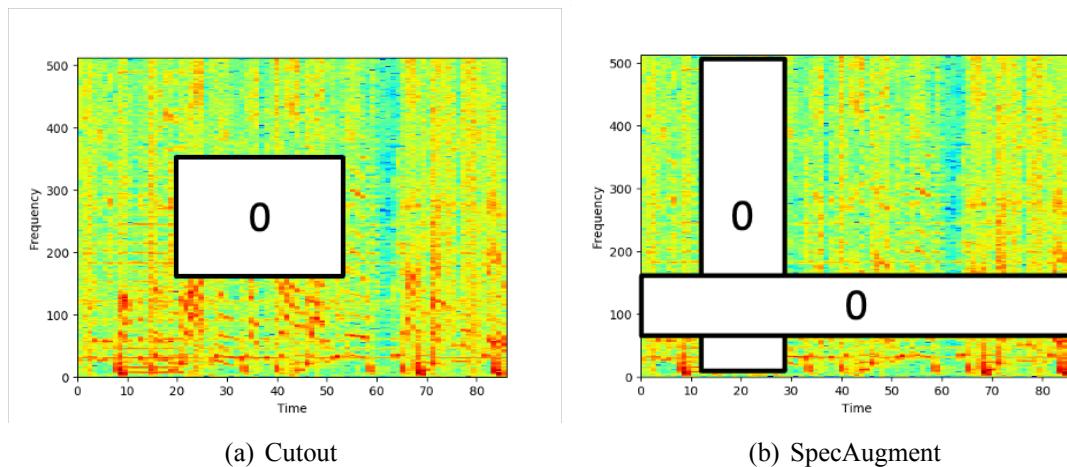


圖 2.6: Cutout 和 SpecAugment

第三章 資料集簡介



3.1 MIREX 資料集

這是台大多媒體資訊檢索實驗室¹內部用來評估音訊指紋效能的資料集 [17]，可分為音樂資料庫和查詢片段兩部分。資料集的內容如表 3.1 所示，資料庫主要為流行歌曲，而查詢片段主要為現實世界的錄音。

	資料庫	查詢片段
音檔類型	GTZAN 資料集、華語和英語歌曲	以手機錄製的音樂片段
音檔格式	單聲道或雙聲道 MP3 檔 多種取樣率，16-bit	單聲道 WAV 檔 44.1 kHz，16-bit
音檔長度	平均約 234 秒	10 秒
數量	10,000 首歌曲	5,692 個片段

表 3.1: MIREX 音訊指紋資料集

3.1.1 清理資料集

由於資料集有重複音樂，因此需要對資料集做清理。重複音樂可分為以下三種：

1. 檔案的二進制內容完全相同
2. 兩首音樂聽起來相同，但二進制內容不同，或者只在品質上有差異
3. 其中一首音樂是另一首音樂的片段

第一種重複音樂可以使用如 MD5 或 SHA1 的雜湊快速找出，在資料集內共找到 499 首音樂和其他音樂發生重複。第二類和第三類重複音樂較難處理，本研究採用音訊指紋的作法，將每個資料庫裡的音樂當作查詢音檔，查詢最相似的音樂。參考 [17] 清理資料集的方式，只刪除有出現在查詢片段中的重複音樂。如此發現 15 首重複音樂，有 11 首屬於第二類，4 首屬於第三類。

¹<http://mirlab.org>

另外也發現參考答案資料有錯誤，有 6 個查詢片段查詢到不在資料集的音樂，因此將 1 首音樂加入資料集來補救。在清理資料集後，資料庫共有 9,487 首音樂，查詢片段不受影響。



3.2 FMA 資料集

FMA 資料集是 Free Music Archive 網站的音樂，可用於多種音樂資訊檢索的任務 [8]。Free Music Archive 大部分的音樂都採用創用 CC (Creative Commons) 授權，因此資料集得以自由的下載和研究。

FMA 資料集共有約十萬首音樂，格式為高品質 MP3 檔案，並包含各種中介資料，如專輯、作者、曲風、標籤... 等等。此資料集共有 161 種曲風，曲風採用樹狀結構，即曲風之下還可以有子曲風，一首歌最多可標記為 3 種曲風。由於資料集龐大，資料集的作者將其分成 4 個不同大小的子集：small、medium、large 和 full，表 3.2 列出這些子集的大小，其中只有 full 包含完整音樂，其他子集都只包含從音樂切出的 30 秒片段。

資料集	數量	曲風個數	音檔長度	大小
small	8,000	8	30 秒	7.4 GiB
medium	25,000	16	30 秒	23 GiB
large	106,574	161	30 秒	98 GiB
full	106,574	161	平均 278 秒	917 GiB

表 3.2: FMA 資料集

3.3 AudioSet

AudioSet [11] 是由 Google 發布的聲音事件資料集，包含 632 個聲音事件分類 (audio event classes) 和 2,084,320 個從 YouTube 影片取出的 10 秒片段，這些 10 秒片段都由人工標記成一個或多個聲音事件分類。聲音事件分類包含了人類和動物聲、音樂曲風、樂器聲、以及日常環境音等類別 [11]。在本研究中，只使用含有「Subway, metro, underground」標籤且沒有 music 或 singing 標籤的子集，約 4.4 小時。



3.4 Aachen Impulse Response

Aachen Impulse Response [15] 是室內殘響的資料集，使用人頭模型來錄製雙耳的室內脈衝響應 (binaural room impulse response)，包含了多種房間大小、音源位置、音源距離、音源方位角的脈衝響應，可用於模擬各種室內環境殘響。資料集曾經過多次的更新，在本研究中使用的資料集版本為 1.4 版，包含 214 個脈衝響應檔案，以 MATLAB 的 MAT 格式儲存，取樣率為 48 kHz，長度從 0.683 秒到 9.987 秒。

3.5 Microphone Impulse Response Project (MicIRP)

Microphone Impulse Response Project [24] 是 Xaudia 發布的麥克風脈衝響應資料集，總共有 69 個脈衝響應，以 24 位元 WAV 檔儲存，取樣率為 44.1 kHz 或 48 kHz，長度從 0.343 秒到 0.5 秒。

3.6 資料集分割

本研究和 NAF 論文使用相同的 FMA 資料集來訓練並測試神經網路，因此仿效 NAF [6] 論文的作法，把 FMA 資料集切割成訓練集、驗證集還有假資料庫。驗證集會用來產生查詢音檔，將在 4.4 節說明產生查詢音檔的方式。假資料庫和驗證集會共同組成查詢測試用的資料庫，而假資料庫的音樂永遠不會是查詢音檔的正確答案，只做為測試資料庫規模用。訓練集、驗證集 1、驗證集 2 以及假資料庫的音樂完全不重複。

- 訓練集：從 fma_medium [8] 中取隨機選出 10000 個 30 秒音樂片段，用於訓練神經網路。
- 驗證集 1：從 fma_medium [8] 中隨機選出 500 個 30 秒音樂片段，用於除了 SVM 整合實驗以外的查詢測試資料，以及 SVM 整合實驗的訓練資料。
- 驗證集 2：從 fma_medium [8] 中隨機選出 500 個 30 秒音樂片段，用於 SVM 整合實驗的查詢測試資料。

- 假資料庫：從 fma_large [8] 中隨機選出 10000 個 30 秒音樂片段，用來填充查詢測試的資料庫。

此外，用於資料擴增的 AudioSet [11]、Aachen [15] 和 MicIRP [24] 資料集，也會按照 8:2 的比例切割成訓練集和驗證集，此設定和 NAF [6] 論文一致。



第四章 研究方法



4.1 音訊指紋系統架構

本論文的實作方法同時使用地標法和深度學習法來辨識音樂，其中地標法是基於 Wang [23] 的辨識方法，以 C++ 語言實作，神經網路法的部分則採用 NAF [6]，以 Python 語言實作。本研究選擇實作地標法來比較傳統演算法和深度學習法的優劣。

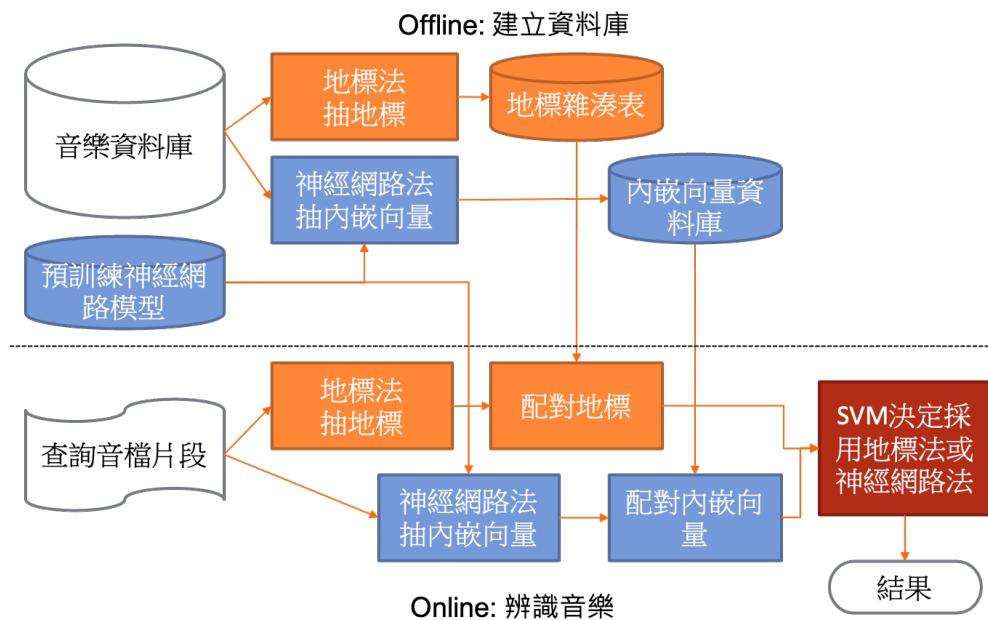


圖 4.1: 音訊指紋系統架構圖

4.2 地標法實作

4.2.1 資料前處理

輸入的音檔會先降低取樣頻率 (downsampling) 到 8,000 Hz，再計算時頻譜。時頻譜的音框大小 (window size) 是 1,024 個取樣點，音框的間距 (hop size) 是 512，也就是音框與音框之間的重疊率為 50%。接下來時頻譜會取對數，計算分貝值，以縮小數值範圍。



4.2.2 抽取地標

地標法的原作 [23] 並沒有說明抽取顯著峰的演算法，因此本研究參考 GitHub 上的 Dejavu 專案 [10] 來實作抽取顯著峰的程式，該專案使用矩形最大值濾波器 (rectangle maximum filter) 來從時頻譜篩選出顯著峰。顯著峰是經過矩形最大值濾波器後，強度仍然不變的時頻圖上的點。這些顯著峰不只是時頻圖上的局部最大值，而且也是周圍矩形區域內的最大值。本研究設定矩形最大值濾波器的時間寬度是考慮的點前後各 5 個音框，共 11 個音框，頻率寬度是考慮的點上下各 10 個 frequency bin，共 21 個 frequency bin。

每個顯著峰按照時間順序，輪流作為錨點，每個錨點在其正時間方向上畫出一個矩形區域，稱為目標區域，則目標區域內的顯著峰可以和錨點配對成地標。將顯著峰配對成地標的參數則是參考唐子翔的論文 [1]。根據唐子翔的研究 [1]，目標區域的時間範圍是錨點的時間加上 5 到 36 個音框，頻率範圍是錨點的頻率 ± 127 個 frequency bin 之間，每個錨點可以在目標區域配對最多 8 個顯著峰，也就是第 2.1.2 節中的 $F = 8$ 。

4.2.3 將地標儲存到雜湊表

在抽取出音樂的地標之後，地標會被轉成雜湊鍵 (hash key) 以及雜湊值 (hash value)，儲存進雜湊表，轉換方式如圖 4.2 所示。一個地標由兩個顯著峰 (t_1, f_1) 、 (t_2, f_2) 組成，其雜湊鍵由三個整數 $(f_1, \Delta f, \Delta t)$ 構成，其中 $\Delta f = f_2 - f_1$ ， $\Delta t = t_2 - t_1$ 。雜湊值則由兩個整數 $(\text{Song ID}, t_1)$ 構成，其中 Song ID 是音樂在資料庫裡的編號。參考唐子翔 [1] 和廖信富 [2] 的論文， f_1 使用 9 個位元， Δf 使用 8 個位元， Δt 使用 7 個位元，Song ID 使用 18 個位元， t_1 使用 14 個位元，因此資料庫可以儲存 $2^{18} = 262,144$ 首音樂。

以上列出的變數中，只有 Δf 可以是正或負整數，其他只能是非負整數。當 $\Delta f < 0$ 時，以二補數 (2's complement) 來儲存負數。 t_1 是地標的時間，以 14 位元儲存，最高可以到 $2^{14} - 1$ ，而 2^{14} 個音框換算成時間大約是 17 分鐘 ($16384 \times 512 \div 8000 \text{ Hz} \div 60$)，若是音樂太長，導致 $t_1 \geq 2^{14}$ ，則 t_1 會取 $\text{mod } 2^{14}$ 。

由於歌曲眾多，每首歌曲所抽取出的地標轉成雜湊後難免會發生雜湊碰撞

(hash collision)，因此本研究採用和廖信富 [2] 相同的方式儲存雜湊表，把雜湊表分成兩個部分，第一個部分是每個雜湊鍵的出現次數，因為雜湊鍵有 24 個位元，所以總共有 $2^{24} = 16,777,216$ 種雜湊鍵，可以用長度 2^{24} 的一維陣列儲存每個雜湊鍵的出現次數。第二個部分是所有的雜湊值，按照雜湊鍵來排序，儲存成一維陣列。

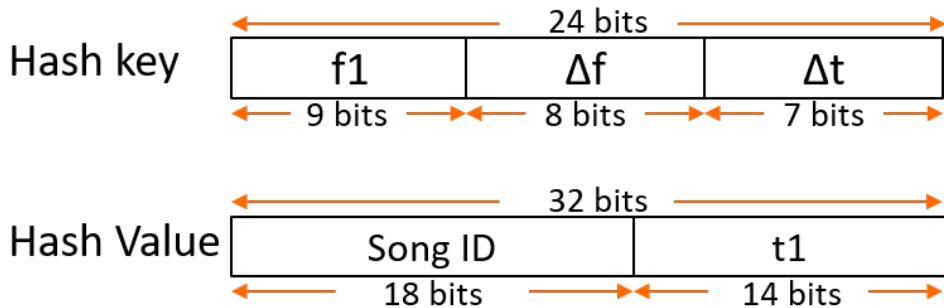


圖 4.2: 地標轉成雜湊鍵和雜湊值的方式

4.2.4 配對地標並計分

系統接收到查詢片段後，會依序使用 4.2.1、4.2.2、4.2.3 節的方法來抽取地標以及轉換成雜湊鍵，但和建立資料庫的不同點在於查詢的階段不需要儲存雜湊表，地標轉換後的雜湊值也不包含歌曲編號，而只是地標的錨點時間 t_1 ，因為查詢時不需要也不知道歌曲編號。

由查詢片段轉換後的雜湊鍵會一一當作索引來查詢資料庫，並取得資料庫雜湊鍵對應的雜湊值 ($\text{Song ID}, t_1$)。為了防止混淆，查詢片段的雜湊值以 t_{query} 表示，資料庫對應的雜湊值則以 $(\text{Song ID}, t_{song})$ 表示。這時查詢結果 $(\text{Song ID}, t_{song})$ 可以解釋成：在查詢片段 t_{query} 的時候歌曲編號 Song ID 已經播放到 t_{song} 的時間，可以用以下公式來反推查詢片段開始時，音樂的播放時間，將此稱為偏移時間 (offset time) δt_k 。

$$\delta t_k = t_{song} - t_{query} \quad (4.1)$$

由於有些音樂很長，會導致 $t_{song} \geq 2^{14}$ ，因此在建立資料庫時，已經先把 t_{song} 取 $\text{mod } 2^{14}$ ，在這一步也會把偏移時間取 $\text{mod } 2^{14}$ ，使得橫跨 $t = 2^{14}$ 的查詢可以正常檢索。

對雜湊表的每個查詢結果計算出偏移時間後，收集所有的 $(\text{Song ID}, \delta t_k)$ ，然後按照歌曲編號來分組，接著對每一首歌曲分別統計每一種偏移時間的出現次數。最高的出現次數，再除以查詢片段的地標個數，就是歌曲的配對分數。在唐子翔的論文中 [1]，會先以歌曲編號來排序，再以偏移時間排序，共兩次的排序來實作出統計次數，但是本研究則是將 $(\text{Song ID}, \delta t_k)$ 依據公式 4.2 轉成一個 32 位元整數值，然後用基數排序法 (radix sort) 排序，因為基數排序法在 CPU 上比較容易實作平行化。

$$value = \text{Song ID} \times 2^{14} + \delta t_k \quad (4.2)$$

4.3 神經網路法實作

本研究以 PyTorch 重新實作 NAF，因此有部分的細節在原 NAF 論文並未提及，也因此音訊指紋辨識的精準度有所差異。神經網路的訓練與測試流程如圖 4.3。

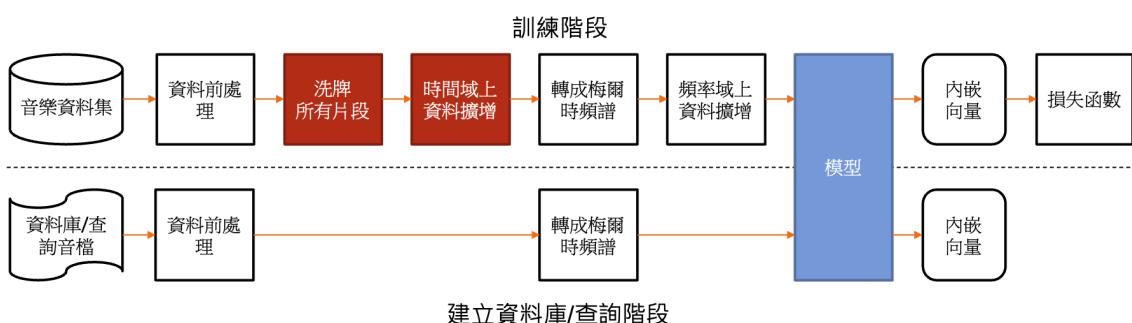


圖 4.3: 神經網路的訓練與測試流程

4.3.1 資料前處理

FMA 資料集和 MIREX 資料集的音樂都是 MP3 格式，因此需要先解碼，本研究採用 FFmpeg 將 MP3 解碼成 16 位元的 WAV 音檔，再將音檔降低取樣頻率 (downsampling) 到 8,000 Hz。神經網路的輸入是 1 秒鐘音訊的對數梅爾時頻譜 (log Mel spectrogram)，將輸入的音檔每 0.5 秒切出 1 秒鐘的重疊音訊片段 (segment)，並將這些片段轉成對數梅爾時頻譜後再個別輸入到神經網路。

為了處理音樂和查詢音量不同的問題，本研究在計算時頻譜之前，先將片段的波形取 L_2 正規化，使片段音量變得一致。其他的論文，如 SpecAugment [18]，是在計算對數時頻譜之後，將時頻譜的平均值正規化成零。採用不同方法的原因是， L_2 正規化會使音量變得一致，而得以保留各個頻率所佔有的能量比例。

4.3.2 模型架構

NAF [6] 的模型架構為表 4.1，可分為卷積編碼器 (convolutional encoder) 和 L_2 投射層 (L_2 projection layer)，神經網路的輸出稱為內嵌向量，會用來做最近鄰居 (nearest neighbor) 搜尋。

卷積編碼器由多層空間可分卷積組成，一層空間可分卷積 (spatially separable convolution)，是由兩次的卷積組成，如圖 4.4，第一次對時頻譜的時間軸做卷積，第二次對頻率軸做卷積，這兩次卷積後面都會加上 Layer normalization [4] 和 ReLU。每一層空間可分卷積會將影像縮小 2 倍，並逐步增加 channel 個數，直到最後一層的輸出是 1×1 的圖像，channel 數為 1024。 L_2 投射層的架構如圖 4.5，將輸入的 1024 維向量拆分成 128 個 8 維子向量，這些子向量會個別經過不同的 2 層全連通層，變成 128 個 1 維數值，然後將這 128 個數值合併成 128 維向量。這 128 維向量還會用 L_2 正規化處理再輸出。

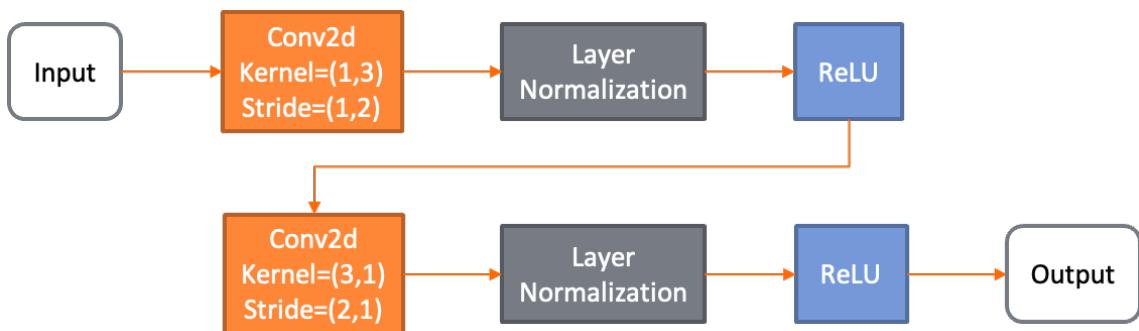


圖 4.4: 空間可分卷積層

4.3.3 資料擴增

本研究使用與 NAF [6] 的論文相同的資料擴增方法，只有資料擴增使用的額外資料集有稍微的不同。



層	輸出大小	output channel
輸入	256×32	1
卷積層 1	128×16	128
卷積層 2	64×8	128
卷積層 3	32×4	256
卷積層 4	16×2	256
卷積層 5	8×1	512
卷積層 6	4×1	512
卷積層 7	2×1	1024
卷積層 8	1×1	1024
拆分成 128 個子向量		8
L_2 投射層 1		32
L_2 投射層 2		1
合併子向量		128

表 4.1: 模型架構

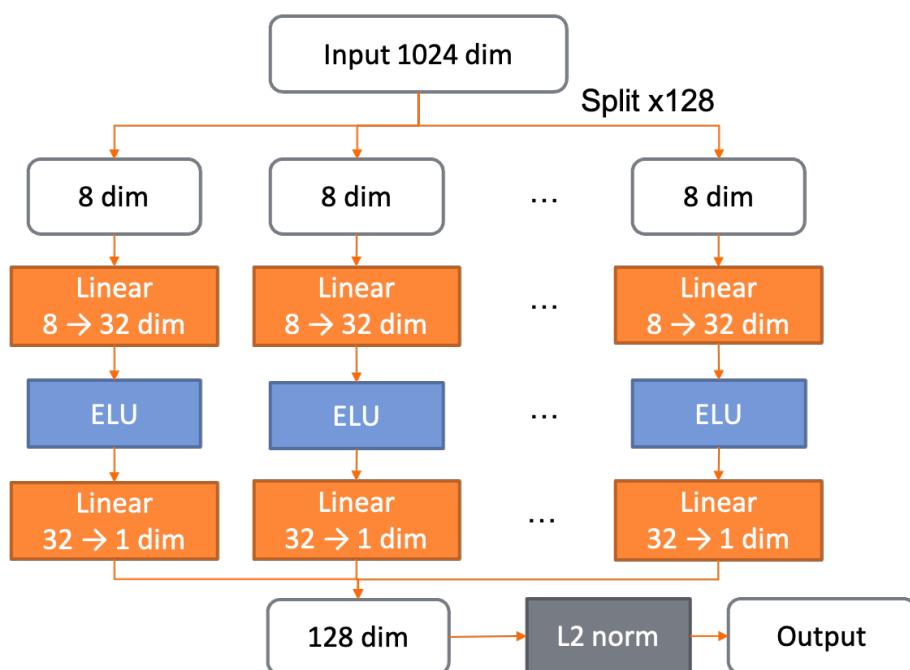


圖 4.5: L_2 投射層



- 隨機時間位移

用來模擬任意的查詢開始時間，做法是從音樂取出 1.2 秒的片段，再從這 1.2 秒的片段隨機選出兩個 1 秒片段作為訓練樣本，其中一個作為原始片段，不加上雜訊和殘響，另一個當作受干擾的查詢，會加上雜訊和殘響。

- 混合背景雜訊

用來模擬查詢時的背景噪音。本研究使用原曲和雜訊之間的 SNR 來衡量雜訊的強度，SNR 的範圍設為 0 到 10dB。NAF 論文使用 AudioSet 資料集和他們自行錄製的餐廳背景噪音作為雜訊資料集，而本研究只使用 AudioSet 資料集作為雜訊。

- 殘響

用來模擬查詢環境的殘響和錄音設備的頻率響應，本研究是將音訊依序和隨機選取的環境殘響和麥克風脈衝響應進行卷積運算。環境殘響為 Aachen 資料集 [15]，麥克風脈衝響應為 MicIRP [24] 資料集。由於殘響檔案過長，造成計算負擔，因此 Aachen 資料集的每個殘響檔案只取前 1 秒。

- Cutout

Cutout 和接下來的 SpecAugment 步驟是使用在時頻譜上，因此在這兩個步驟之前，會先把音訊轉成對數梅爾時頻譜。Cutout 會把時頻譜上的一個隨機矩形區域填入零，如圖 2.6 (a)，矩形的長寬各是時頻譜長度和寬度的 1/10 到 1/2。在同一個 batch 裡的每個訓練樣本都使用同樣大小和位置的矩形。

- SpecAugment

SpecAugment 會在時頻譜上隨機的畫出橫向遮罩和縱向遮罩，並將遮罩內的值填入零，如圖 2.6 (b)。SpecAugment 的橫向遮罩和縱向遮罩的大小各為時頻譜高度和寬度的 1/10 到 1/2。在同一個 batch 裡的每個訓練樣本都使用同樣大小和位置的遮罩。

4.3.4 損失函數 (loss function)

NAF [6] 模型在訓練時使用了對比學習 (contrastive learning) 的 NT-Xent 損失函數 [7]，對比學習的 mini-batch 是 $N/2$ 個訓練資料組成，經由資料擴增後，每個

訓練資料會產生兩個訓練樣本 (s_{org}, s_{aug})，共 N 個訓練樣本。NT-Xent 損失函數會學習同一個 mini-batch 中任兩個訓練樣本的相似性，定義相似為從同一個訓練資料產生的兩個變種，不相似為從不同訓練樣本產生的兩個變種，則損失函數會使相似的兩個變種的內嵌向量較為靠近，不相似的兩個變種的內嵌向量則遠離。

NT-Xent 損失函數 [7] 如式 4.4 所示，是由每個正樣本的 ℓ 函數取平均。式 4.3 的 ℓ 函數則可以解釋成計算 (i, j) 的 softmax，而式子中的 τ 是 softmax 的溫度參數，在本研究都設為 0.05。

$$\ell(i, j) = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^N \mathbb{1}(k \neq i) \exp(\text{sim}(z_i, z_k)/\tau)} \quad (4.3)$$

$$L = \frac{1}{N} \sum_{k=1}^{N/2} [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \quad (4.4)$$

sim 函數是兩個樣本 (z_i, z_j) 的相似度，NAF [6] 使用餘弦相似度來作為相似度衡量方式，因為神經網路的輸出有經過 L_2 正規化，所以餘弦相似度可以轉為向量內積。

$$\text{sim}(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\|_2 \|z_j\|_2} = z_i^T z_j \quad (4.5)$$

4.3.5 訓練方法

訓練開始前，會先把訓練資料前處理成 8000Hz、單聲道的音檔，再把音檔每 0.5 秒切出一個 1.2 秒片段，接著才能做訓練。NAF 模型的訓練流程包含以下幾個步驟：

1. 隨機從訓練資料取出 batch size/2 個音樂片段，取後不放回。
2. 將這些音樂片段，以資料擴增的方式產生出 batch size 個訓練樣本 $\{x_1^{org}, x_1^{aug}, \dots, x_{N/2}^{org}, x_{N/2}^{aug}\}$ 。
3. 輸入神經網路得到 $\{z_1^{org}, z_1^{aug}, \dots, z_{N/2}^{org}, z_{N/2}^{aug}\}$ ，利用 NT-Xent Loss 來訓練神經網路。



以上的步驟是訓練過程的一步 (step)，當整個訓練資料都使用過一次時，稱為一個 epoch。以下是模型訓練使用的超參數 (hyperparameter)：

- Batch size: 640 (320 個成對樣本)
- Learning rate: 10^{-4}
- Optimizer: Adam
- Learning rate scheduler: Cosine Decay 到 10^{-7} ，不採用 warmup 和 restart
- Epochs: 100

梅爾時頻譜的參數是：

- STFT window size: 1024
- STFT hop size: 256
- 梅爾頻率範圍：300~4000Hz
- 梅爾時頻譜大小 ($F \times T$)： 256×32

4.3.6 將內嵌向量儲存到資料庫

在訓練完神經網路後，就可以將資料庫音樂每 0.5 秒切出 1 秒片段，輸入到神經網路，取得內嵌向量並存到用於最鄰近搜尋 (nearest neighbor search) 的資料庫。神經網路法抽出的特徵數量固定是每 0.5 秒一個，因此一首歌曲的第 t 個內嵌向量是從 $0.5t$ 秒處的一秒片段抽出的。可以將每一首音樂按照音樂編號順序儲存其內嵌向量。設第 $SongID$ 首音樂有 n_{SongID} 個片段，則可以將這首音樂的第 t 個片段的內嵌向量分配到片段編號

$$\left(\sum_{i=0}^{SongID-1} n_i \right) + t \quad (4.6)$$

使得每首音樂的每個片段都有不同的編號。將片段編號轉換回音樂編號和時間則可以採用二分搜尋法，因為內嵌向量已經依照音樂編號排序。

由上可知，資料庫需要儲存訓練好的模型、所有音樂的內嵌向量以及每首音樂的片段數量 n_i 。模型的大小是固定的，在本研究中都是約 65 MiB，而儲存每首音樂的 n_i 只需要為每一首音樂存放一個 4-Byte 整數，基本上可忽略，但是所有音樂的內嵌向量會佔用極大的空間，是內嵌向量的總數乘上一個內嵌向量的大小。一個內嵌向量是 128 維向量，若使用單精度浮點數儲存，則佔用 512 Bytes=0.5 KiB，而每分鐘會抽出 $60/0.5 = 120$ 個內嵌向量，佔用 60 KiB，超出 MIREX 比賽規定的每分鐘 50 KB [17]，因此壓縮資料庫有其必要性。

另外，搜尋速度也會成為問題，雖然在低維度向量情況下，最近鄰居搜尋可以使用 k-d tree 加速，但是內嵌向量的維度是 128，維度過大，k-d tree 搜尋會退化成暴力搜尋，而且事實上在本研究不需要完全正確的搜尋結果，而只需要近似解。因此，本研究採用和 NAF [6] 論文相同的資料庫索引方式，是倒排索引 (inverted file) 加上乘積量化 (product quantization)。資料庫索引的實作採用開源的 Faiss [16] 套件。

倒排索引是一種資訊檢索的索引方式，用於最鄰近搜尋時，是將向量分成 C 個群，使得可以只搜尋一部份的分群就找到距離最近的向量，達到加速搜尋的目的，如圖 4.6。做法是在搜尋時，計算查詢向量到每個群的中心的距離，然後只從最靠近的 τ 個群裡搜尋最鄰近向量。NAF [6] 論文採用的分群數 C 是 200，但沒有提到 τ 的值，因此本研究設定 τ 為 50，也就是搜尋約 1/4 的資料庫。

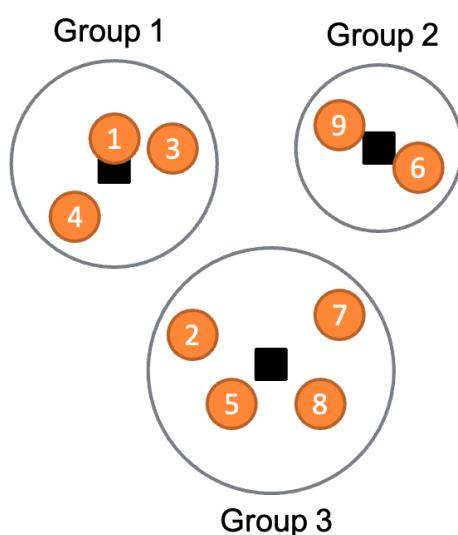


圖 4.6: 倒排索引

乘積量化是一種量化壓縮方式，用來壓縮最鄰近搜尋的向量資料庫。做法是，

對於資料庫裡的 d 維向量，將其拆成 M 個子向量，接著對這些子向量分別做量化，量化的方法是把子向量以 K-means 分成 2^b 個群，因此一個子向量就可以量化成 b 個位元。本研究採用和 NAF [6] 論文相同的參數： $M = 64$ 、 $b = 8$ 。

倒排索引可以和乘積量化同時使用，此時乘積量化所量化的向量可以是原始向量減去倒排索引的分群中心後的殘差，殘差的向量長度比原向量還小，使得量化誤差可以縮小。

4.3.7 查詢與計分方式

查詢方式參考前人的論文 [12] [6]，流程圖如圖 4.7。神經網路法接收到查詢音檔時，首先經過前處理，得到 L 個片段，將這些片段輸入神經網路得到序列 q_0, q_1, \dots, q_{L-1} 。接著對於每個 q_i ，從資料庫裡搜尋最靠近 q_i 的 k 個鄰居的片段編號 $I_{ij}, j = 1 \sim k$ ，並使用第 4.3.6 節的方法把片段編號轉回音樂編號和時間 $(SongID_{ij}, t_{ij})$ 。接著為了把時間統一到查詢開始的時間，把 t_{ij} 減去 i 得到 $(SongID_{ij}, t_{ij} - i)$ 。令 S 是所有的 $(SongID_{ij}, t_{ij} - i)$ 構成的集合，代表可能歌曲的候選者。對於每個 $(song_m, t_m) \in S, m = 1 \sim |S|$ ，從資料庫取出音樂 $song_m$ 的第 t_m 個片段開始的連續 L 個片段，和查詢的 q_0, \dots, q_{L-1} 序列逐一計算餘弦相似度（同內積），再取平均，做為 $(song_m, t_m)$ 的得分。若取出的音樂片段超出音樂的範圍，則超出的部分相似度為零。同一首 $song_m$ 的 $(song_m, t_m)$ 的最高分即為音樂的配對分數，而最高分的 $song$ 即為查詢結果。

和 NAF [6] 的不同點是，NAF 沒有考慮到檢索到的片段編號再加上查詢片段的長度後，有可能會超出原本片段編號所屬音樂的範圍，使得接下來用於計分分配對的序列混合了兩首音樂的序列。本研究的作法考慮到音樂編號，在配對序列超出音樂範圍的部分，相似度為零，較符合實際搜尋情況。另外，為了方便接下來的整合模型，本研究使用平均分數，使配對分數介在 -1 到 1 之間，而不是 NAF [6] 採用的總分。

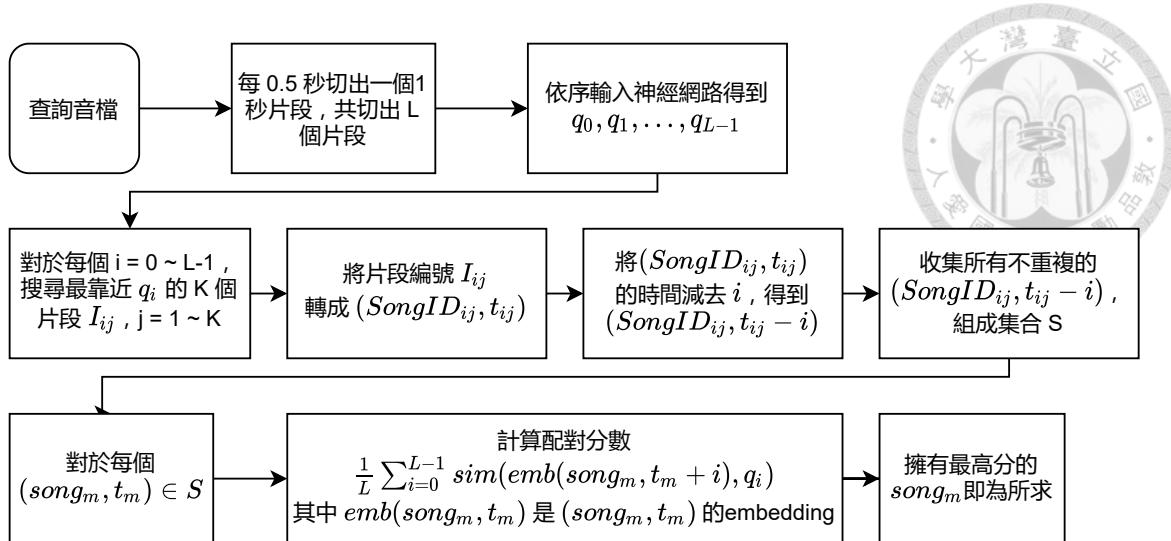


圖 4.7: 查詢與計分方式流程圖

4.4 評量指標

本論文評量方法為第一名的正確率 (top-1 hit rate)，會以兩種資料集，FMA 和 MIREX 資料集來做評估。測試資料包含資料庫音樂和查詢音檔，為了限縮研究範圍，所有的查詢音檔都是 10 秒。

- FMA：由於 FMA 資料集 [8] 不是針對音訊指紋設計的，因此需要先產生查詢音檔。查詢音檔由第 3.6 節提到的驗證集的 500 首音樂，以類似先前提到的資料擴增方式來產生 2000 個查詢，具體的說，查詢音檔的產生共有三個步驟：

1. 隨機時間位移：從音樂中隨機選出一個連續 10 秒的片段。
2. 混合背景雜訊：隨機從 AudioSet [11] 中選出音檔充當背景雜訊，再將音樂片段和背景雜訊以指定的 SNR 來混合。
3. 殘響：從 Aachen 資料集 [15] 和 MicIRP [24] 資料集隨機各取出一個脈衝響應，然後將查詢音檔依序套用這兩個脈衝響應。

下一章的實驗會以不同 SNR 的雜訊產生的查詢音檔來測試模型精準度，以便比較模型處理各種程度雜訊的能力，本研究用以測試的 SNR 是 $\{-6, -4, -2, 0, 2, 4, 6, 8\}$ dB。

FMA 資料集的音樂資料庫則是由第 3.6 節提到的驗證集的 500 首音樂，再加上假資料庫的 10000 首音樂構成。驗證集有兩個，在改進實驗會使用驗證集 1 來做測試，但是在 SVM 整合實驗則是把驗證集 1 當作是 SVM 的訓練資料，驗證集 2 才是測試資料，這是因為 SVM 也需要訓練，不能和神經網路使用一樣的訓練集。

- MIREX：使用 MIREX 資料集的音樂資料庫及查詢音檔直接測試。

4.5 改進實驗

4.5.1 二階段洗牌 (Two-Phase Shuffling)

原本神經網路的訓練資料是把音樂的所有 1 秒片段全部打亂 (shuffle)，但是如此的訓練流程導致訓練時讀取資料集的行為呈現隨機讀取。若依照 NAF 論文的說法，訓練 100 個 epoch 需要 30 個小時 [6]，則由於每個 epoch 需要讀取所有訓練音樂片段，訓練音樂有 10000 首，每首約 30 秒，可切出 59 個片段，因此訓練時每秒需要讀取 $100 \times 10000 \times 59 / (30 \times 3600) \approx 546$ 個片段。一般的硬碟無法處理如此大量的隨機讀取要求，以至於 NAF [6] 的程式要求資料集需要存放在固態硬碟上才能訓練。為了解決硬碟 I/O 瓶頸，本研究提出先打亂音樂，再打亂片段的做法，稱為二階段洗牌，如圖 4.8。這個做法有一個參數 *shuffle_size* 可以調整。首先，先打亂所有訓練音樂的順序，接著，把打亂過的音樂每 *shuffle_size* 首分為一組，每個分組各自打亂分組內音樂的片段，最後按照打亂後片段的順序來切出 mini-batch。如此只要 RAM 可以儲存 $2 \times \text{shuffle_size}$ 首音樂，用來打亂片段，就只需要每個 epoch 讀取每首音樂一次，約每秒讀取 $100 \times 10000 / (30 \times 3600) \approx 9.3$ 次檔案，理論上可以改善 I/O 效能。

在實驗過程中，發現二階段洗牌能增加精準度，然而由於訓練用機器的 I/O 足夠快，因此 I/O 不是瓶頸，二階段洗牌實際上不會改變訓練速度。本論文將在下一章比較 *shuffle_size* 參數為 1、10、100、1000 的情形，並和不使用二階段洗牌的結果做比較。

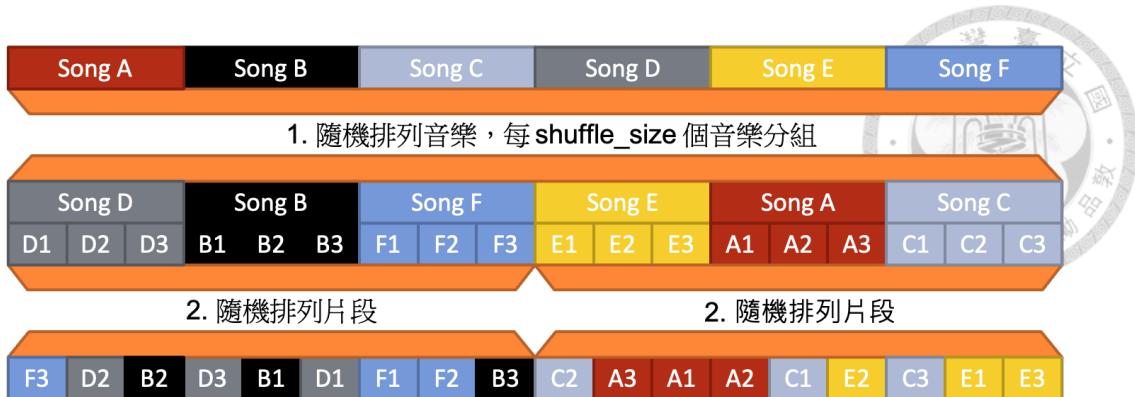


圖 4.8: 二階段洗牌

4.5.2 資料擴增改良

雖然 NAF 使用資料擴增來模擬查詢時的干擾，但是其仍有部分設定和真實查詢情況不吻合，不能合理的模擬查詢，因此本研究提出幾個對資料擴增流程的改良實驗。

隨機時間位移

原先 NAF 論文的隨機時間位移是先從音樂取出 1.2 秒的片段，再從 1.2 秒的片段隨機取出兩個一秒片段，如此兩個片段的時間位移最多為 0.2 秒。然而原始設定中，原曲和查詢片段都是每 0.5 秒切出一個 1 秒片段，再輸入到神經網路，由於查詢開始時間可以是原曲的任意處，這導致在最糟情況下，原曲和查詢音檔切出的片段會相差 0.25 秒，如圖 4.9，超過訓練時可能遇到的最大時間位移，使得精準度較低。

因此本研究會將最大時間位移改為 0.25 秒，也就是先從音樂取出 1.25 秒的片段，再從這 1.25 秒片段隨機切出兩個 1 秒片段。本研究會在實驗中比較原始設定 0.2 秒隨機時間位移，和 0.25 秒隨機時間位移的精準度。為了得知隨機時間位移的重要性，本研究也測試移除隨機時間位移後的結果，實驗結果將在第 5 章討論。

背景雜訊

原本 NAF 論文在資料擴增中的背景雜訊步驟，加入雜訊後的 SNR 在 0 到 10 分貝之間，但是聆聽過加入雜訊的 FMA 音樂，以及 MIREX 的查詢音檔後，我們

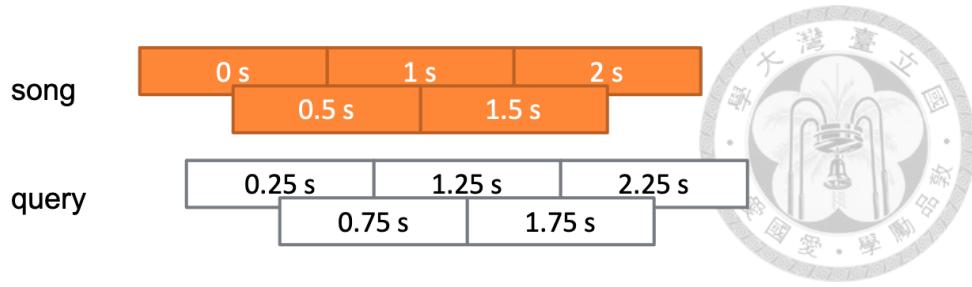


圖 4.9: 查詢和原曲相差 0.25 秒的情況，片段內的數字表示對應到原曲的時間

認為 MIREX 查詢資料集遭受到嚴重的背景雜訊干擾，可能有多數查詢的 SNR 小於 0dB，而且也有些查詢難以聽出裡面的音樂。為了處理 MIREX 資料集過於嚴重的雜訊，本研究會實驗將背景雜訊 SNR 的範圍改為 -5~10dB 以及 -10~10dB，並觀察精準度的變化。

殘響

在音訊指紋檢索的時候，經常會有殘響的問題，而且錄音搜尋音樂通常不會從音樂的一開始就搜尋，所以可以在錄音的一開始聽到錄音開始前音樂產生的殘響。由於神經網路的輸入只有 1 秒的片段，因此片段開始前的音樂造成的殘響佔有更大的比例，有時錄音片段的前半段大多來自更早之前音樂的殘響。為了更真實的模擬殘響，本研究會在訓練資料取出的每個 1 秒片段往前多取 1 秒，將這 2 秒的音訊連接起來，加入雜訊和殘響，然後刪除開頭 1 秒的音訊。

4.5.3 對查詢做多次時間位移

本方法是來自廖珮好的論文 [3]，該論文提出地標法的精準度改良方法，而本論文修改此方法以套用到神經網路法上。據我們所知，本論文是第一篇將此方法套用在神經網路音訊指紋的論文。

神經網路法在建立資料庫以及檢索時，輸入音檔每 0.5 秒切出一個 1 秒片段，再輸入到神經網路，但是原曲和查詢音檔的時間差可能不是 0.5 秒的倍數，使得切出的片段沒有對齊，如圖 4.10。雖然神經網路在訓練時，有使用資料擴增的方法讓神經網路適應 0.2 秒的時間差，但是如果多次的對查詢音檔做時間位移，則可以讓原曲和查詢音檔的時間差縮小，也許可以增加正確答案的配對分數，使得精準度提高。

本項實驗定義參數 N ，表示對同一查詢音檔重複查詢的次數，在廖珮好的論



文 [3] 中，多次時間位移的方法是，從查詢音檔中抽取出地標，再將查詢音檔每次位移音框大小的 $1/N$ ，如此的步驟重複 N 次後，把所有抽出的地標收集起來搜尋資料庫作為查詢結果。

然而在神經網路法中，沒有地標特徵，因此本研究修改為查詢 N 次，每次查詢完，就把查詢音檔位移 $0.5/N$ 秒。在重複 N 次查詢後，查詢音檔正好會被位移 0.5 秒，也就是跳過一個片段。接著計算這 N 次查詢中的最高分作為查詢結果。

本實驗會嘗試 $N = 1$ 、 $N = 2$ 和 $N = 4$ ，並比較精準度和查詢時間。用在地標法時，根據廖珮好的論文 [3]，baseline 設定為 $N = 4$ ，而用在神經網路法時，baseline 設定為 $N = 1$ 。

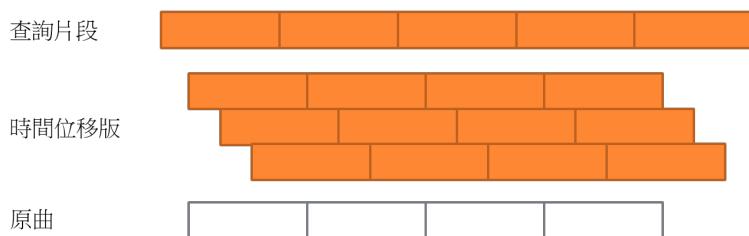


圖 4.10: 對查詢做多次時間位移

4.6 以 SVM 整合地標法和神經網路法的結果

在實驗中，發現到地標法和神經網路法各有能夠處理的查詢和不能處理的查詢，為了讓精準度能夠再提升，本研究先用這兩種方法各計算出配對到的音樂和配對分數，接下來把配對分數輸入到 SVM 模型，讓 SVM 預測何種做法的答案最可能正確，即可整合兩種方法的結果。

SVM 模型的輸入有兩個，是地標法和神經網路法的配對分數，輸出是這兩種方法中，最可能正確的方法。訓練資料是驗證集 1，以 SNR -10 到 10dB 產生 2000 個 10 秒查詢音檔，再篩選使用地標法和神經網路法後，恰有一種方法答對的查詢。訓練資料的標籤則是答對的方法。測試查詢是從驗證集 2 產生，測試用資料庫是驗證集 2 加上假資料庫。

此實驗將會在改進實驗完成後進行，並測試兩種不同的 SVM 核函數 (kernel function)：線性 SVM 和徑向基函數核 (radial basis function，簡稱 RBF)。SVM 實作採用開源的 Scikit-learn 套件 [19]。



4.7 實驗環境

本研究使用國網中心的台灣杉二號來訓練神經網路模型，台灣杉二號共有 252 個計算節點，其單一節點規格如表 4.2 所示¹。台灣杉二號使用 Nvidia 的 Volta 架構顯示卡 (GPU)，可以利用混合精度訓練 (mixed precision training) 來加速訓練。混合精度訓練只需要 19 小時，相較之下，不使用混合精度訓練需要 29 小時。本研究的每個模型，都只使用一張 GPU 和 4 個中央處理器 (CPU) 核心來訓練。

CPU	Intel Xeon Gold 6154 18 核心 3.0 GHz × 2
GPU	NVIDIA Tesla V100 × 8 (32 GB 記憶體)
RAM	768 GB

表 4.2: 訓練機器規格

訓練完模型之後，會把訓練的模型下載到實驗室的電腦，並在實驗室的電腦上建立音訊指紋資料庫和查詢測試，SVM 模型的訓練也會使用實驗室電腦。實驗室的電腦規格如表 4.3 所示。

CPU	Intel Core i7-6800K CPU 6 核心 3.40 GHz
GPU	NVIDIA GeForce GTX 1080 Ti × 2 (11 GB 記憶體)
RAM	96 GB
HDD	WD My Book 8 TB
SSD	Intel SSD 540s 240 GB 和 Crucial BX500 2 TB

表 4.3: 測試機器規格

¹https://iservice.nchc.org.tw/nchc_service/nchc_service_twcc.php

第五章 實驗結果探討

本研究會進行一連串的實驗，各實驗決定的參數會套用到下一個實驗，因此接下來各實驗結果的表格，會以粗體字表示最後選擇的實驗參數。另外，SVM 整合實驗會在實驗一到實驗五完成之後才進行，並使用這些實驗決定的參數。實驗最終得到的神經網路模型，其執行時間的效能分析可以參考附錄。

5.1 基礎神經網路模型與地標法之比較

在對神經網路做改進前，首先來觀察和傳統作法的精準度差異。由圖 5.1 可發現，以 FMA 音樂資料集測試時，當雜訊 $\text{SNR} \leq -2\text{dB}$ 時，地標法的精準度較高，而當雜訊 $\text{SNR} \geq 0\text{dB}$ 時，神經網路的精準度較高，所以地標法較能處理低 SNR 的吵雜環境，而神經網路較能處理高 SNR 的安靜環境。而觀察 MIREX 資料集精準度，發現地標法的精準度較高，根據對資料集的觀察，我們認為是因為 MIREX 查詢音檔的錄音環境非常吵雜。

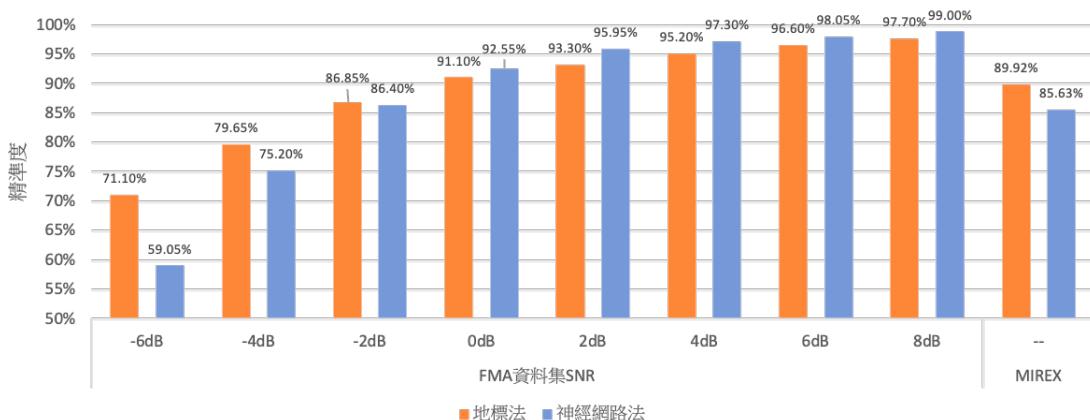
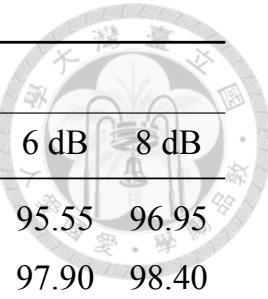


圖 5.1: 基礎模型與地標法的精準度比較圖

方法	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
地標法	71.10	79.65	86.85	91.10	93.30	95.20	96.60	97.70	89.92
神經網路法	59.05	75.20	86.40	92.55	95.95	97.30	98.05	99.00	85.63

表 5.1: 基礎模型與地標法的精準度 (單位：%)



方法	FMA 驗證集 2 雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
地標法	70.65	79.65	85.40	89.80	92.25	94.35	95.55	96.95
神經網路法	58.10	74.30	86.50	92.40	95.30	96.90	97.90	98.40

表 5.2: 基礎模型與地標法的精準度 (續) (單位: %)

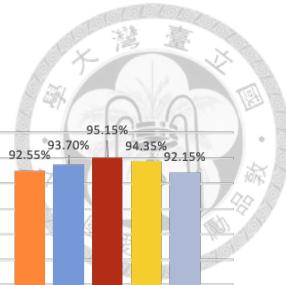
資料庫的大小如表 5.3 所示，此大小已排除掉資料庫中的音樂名稱列表，因為音樂名稱列表的大小和執行環境有關。由圖表可知，雖然神經網路法需要儲存模型，但是神經網路法所使用的空間還是遠比地標法小。

測試資料集	方法	資料庫大小
FMA	地標法	329.47 MiB
	神經網路法	106.96 MiB
MIREX	地標法	1903.43 MiB
	神經網路法	367.58 MiB

表 5.3: 神經網路法與地標法的資料庫大小

5.2 實驗一：二階段洗牌

二階段洗牌有一個可調的超參數 $shuffle_size$ ，表示每次隨機排序音樂片段時使用的音樂數量。雖然 Baseline 並未使用二階段洗牌，但是由於訓練資料共 10000 首音樂，因此當 $shuffle_size$ 為 10000 時，即表示第二階段的洗牌將會隨機排列所有 10000 首音樂的所有片段，而和 Baseline 的洗牌步驟一致，可將 Baseline 視同 $shuffle_size = 10000$ 。二階段洗牌的實驗結果如表 5.4、表 5.5、圖 5.2 與圖 5.3。由圖 5.2 和圖 5.3 可發現，無論在 FMA 還是在 MIREX 資料集上，都是 $shuffle_size$ 為 100 時表現最好。當 $shuffle_size$ 減小，則第二階段的洗牌一次只洗牌較少個音樂片段，因此屬於同一首音樂的片段更容易集中在一個 batch 裡。而我們認為，從同一首音樂取出的兩個不同片段，比取自不同音樂的兩個片段還要更為相似，對神經網路來說，較難分辨出來，所以 $shuffle_size$ 減少時，對神經網路的挑戰較大。適當的降低 $shuffle_size$ 可以提升精準度，但是降低太



多，則神經網路難以學習，反而使精準度降低。

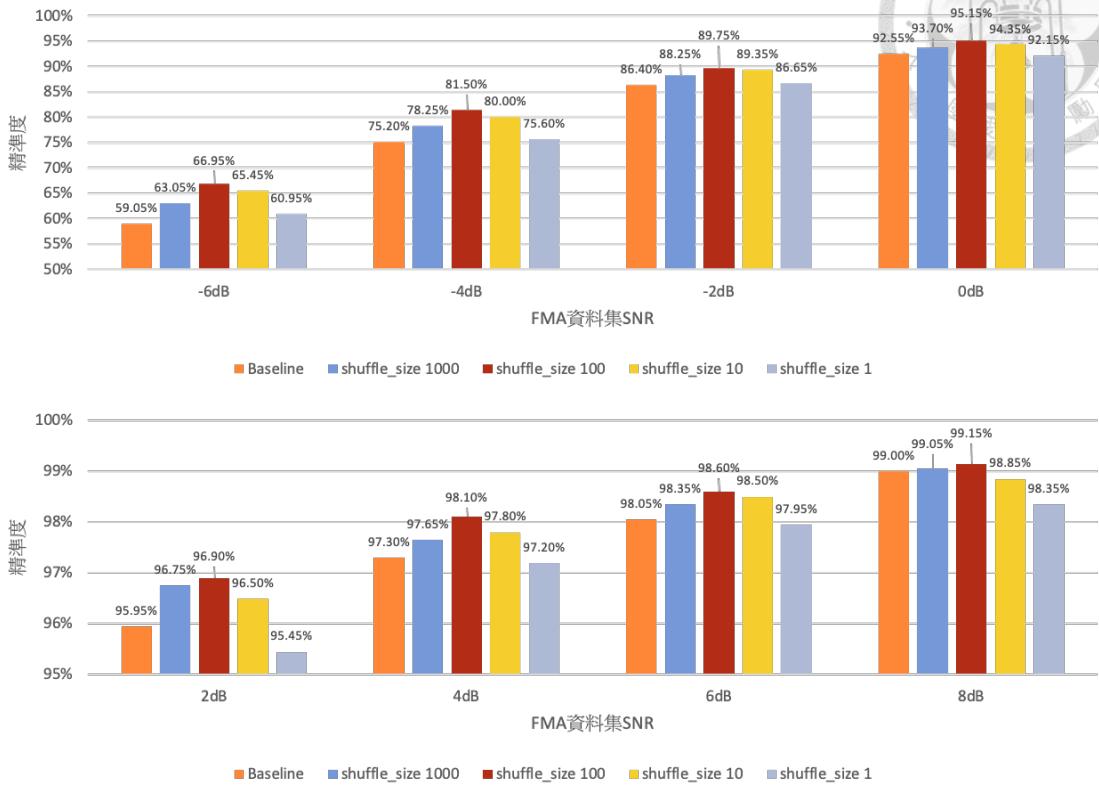


圖 5.2: 二階段洗牌實驗在 FMA 驗證集 1 上的精準度

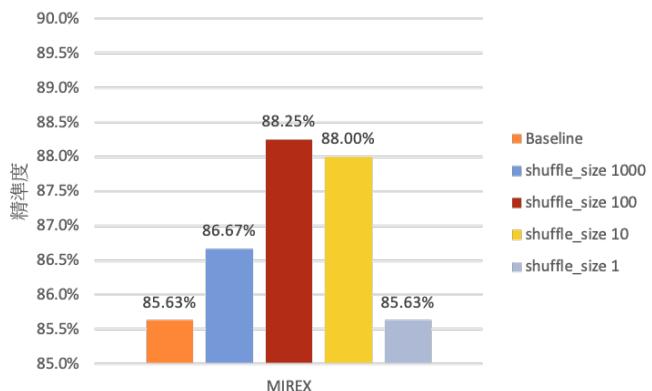


圖 5.3: 二階段洗牌實驗在 MIREX 資料集上的精準度



shuffle_size	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
無 (baseline)	59.05	75.20	86.40	92.55	95.95	97.30	98.05	99.00	85.63
1000	63.05	78.25	88.25	93.70	96.75	97.65	98.35	99.05	86.67
100	66.95	81.50	89.75	95.15	96.90	98.10	98.60	99.15	88.25
10	65.45	80.00	89.35	94.35	96.50	97.80	98.50	98.85	88.00
1	60.95	75.60	86.65	92.15	95.45	97.20	97.95	98.35	85.63

表 5.4: 二階段洗牌實驗精準度 (單位 : %)

shuffle_size	FMA 驗證集 2 雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
無 (baseline)	58.10	74.30	86.50	92.40	95.30	96.90	97.90	98.40
1000	61.10	77.15	88.05	93.20	95.70	97.35	98.00	98.55
100	65.55	81.15	89.95	94.40	96.35	97.70	98.45	98.85
10	65.35	79.30	88.75	93.65	96.25	97.00	98.10	98.45
1	59.55	73.70	85.25	92.25	94.80	96.40	97.35	98.05

表 5.5: 二階段洗牌實驗精準度 (續) (單位 : %)



5.3 實驗二：改良資料擴增-隨機時間位移

隨機時間位移的實驗結果如圖 5.4、圖 5.5、表 5.6、以及表 5.7。從圖中可以看出，當訓練時不使用隨機時間位移的資料擴增，也就是隨機時間位移 0 秒，則精準度會大幅下降。由於輸入到神經網路的音訊被切成每 0.5 秒一個，因此查詢片段和原曲的片段，即使是最靠近的情況下，仍有可能相差 0.25 秒。而 NAF [6] 的神經網路訓練時，隨機時間位移最多只有 0.2 秒，可能無法涵蓋所有的時間位移。若把隨機時間位移改成 0.25 秒，則可能因為在訓練時涵蓋所有的時間位移，使精準度提高，但是也可能因為訓練時沒有容錯範圍，使得模型難以訓練。準確度下降。觀察 FMA 資料集的精準度，發現當雜訊很大 ($SNR \leq -2dB$) 時，將隨機時間位移改為 0.25 秒可以提高精準度。但是當雜訊較小，則精準度的提升變少，有時精準度還會下降。觀察 MIREX 資料集的精準度，則發現隨機時間位移 0.2 秒和 0.25 秒的精準度幾乎一樣，沒有改進。

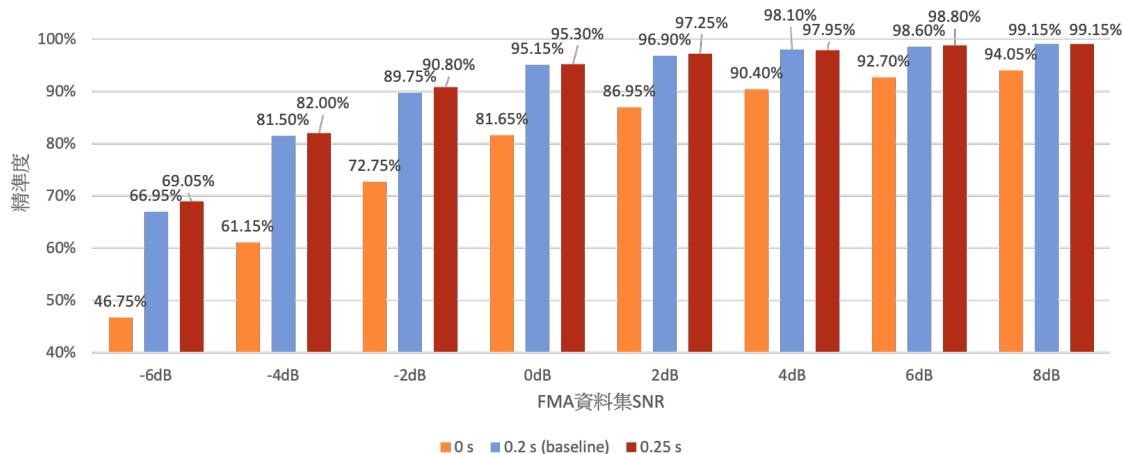


圖 5.4: 隨機時間位移實驗在 FMA 驗證集 1 上的精準度

時間位移大小	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
0 s	46.75	61.15	72.75	81.65	86.95	90.40	92.70	94.05	77.67
0.2 s (baseline)	66.95	81.50	89.75	95.15	96.90	98.10	98.60	99.15	88.25
0.25 s	69.05	82.00	90.80	95.30	97.25	97.95	98.80	99.15	88.28

表 5.6: 改良時間位移資料擴增實驗精準度 (單位：%)

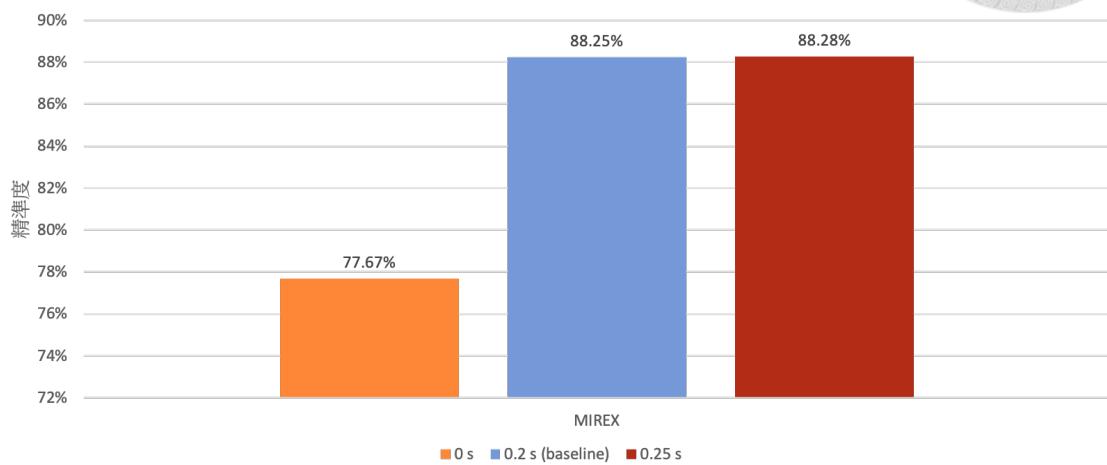


圖 5.5: 隨機時間位移實驗在 MIREX 資料集上的精準度

時間位移大小	FMA 驗證集 2 雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
0 s	45.45	60.65	73.65	83.40	88.30	91.25	93.60	94.60
0.2 s (baseline)	65.55	81.15	89.95	94.40	96.35	97.70	98.45	98.85
0.25 s	67.85	81.90	90.05	94.30	96.25	97.70	98.25	98.85

表 5.7: 改良時間位移資料擴增實驗精準度 (續) (單位 : %)



5.4 實驗三：改良資料擴增-雜訊強度

雜訊強度的實驗結果如圖 5.6、圖 5.7、表 5.8、以及表 5.9。由表 5.8 可知，如果訓練時沒有加雜訊的資料擴增，精準度會降到非常低，所以雜訊是必須要有的資料擴增。將資料擴增雜訊 SNR 改成 $-5 \sim 10\text{dB}$ ，在 FMA 測試資料上只有 $\text{SNR} \leq 0\text{dB}$ 的查詢精準度有提高， $\text{SNR} \geq 2\text{dB}$ 的查詢精準度卻下降。而將資料擴增雜訊 SNR 改成 $-10 \sim 10\text{dB}$ 時，相對於資料擴增雜訊 SNR $-5 \sim 10\text{dB}$ 的設定，在 FMA 測試資料上只有 $\text{SNR} \leq -4\text{dB}$ 的查詢精準度有提高，其他則持平或下降。根據對資料集的檢驗，我們認為當訓練時雜訊變強的時候，神經網路會傾向於把更多的訊號視為雜訊，因此遇上雜訊較輕微的查詢時，可能會把音樂視為雜訊忽略掉，而且人工檢查也發現到，這些因訓練時雜訊變強而辨識錯誤的查詢，通常都聽不出包含的音樂。

在 MIREX 資料集上，則是資料擴增雜訊的 SNR 下限越小，精準度越高，因為 MIREX 資料集的查詢片段大多遭受了嚴重的雜訊干擾。

資料擴增雜訊 SNR	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
無雜訊	36.05	49.45	61.75	72.70	81.70	88.95	93.05	95.90	82.50
0 ~ 10 dB (baseline)	69.05	82.00	90.80	95.30	97.25	97.95	98.80	99.15	88.28
-5 ~ 10 dB	76.15	85.40	92.10	95.85	96.95	98.05	98.55	99.05	90.46
-10 ~ 10 dB	77.90	86.85	92.00	95.30	96.75	97.95	98.65	99.10	91.99

表 5.8: 改良雜訊資料擴增實驗精準度 (單位 : %)

資料擴增雜訊 SNR	FMA 驗證集 2 雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
無雜訊	37.15	50.40	62.50	72.45	80.35	86.85	90.85	93.70
0 ~ 10 dB (baseline)	67.85	81.90	90.05	94.30	96.25	97.70	98.25	98.85
-5 ~ 10 dB	75.40	86.65	91.90	95.10	96.90	97.55	98.10	98.80
-10 ~ 10 dB	78.30	87.20	91.70	94.35	96.45	97.65	98.20	98.75

表 5.9: 改良雜訊資料擴增實驗精準度 (續) (單位 : %)

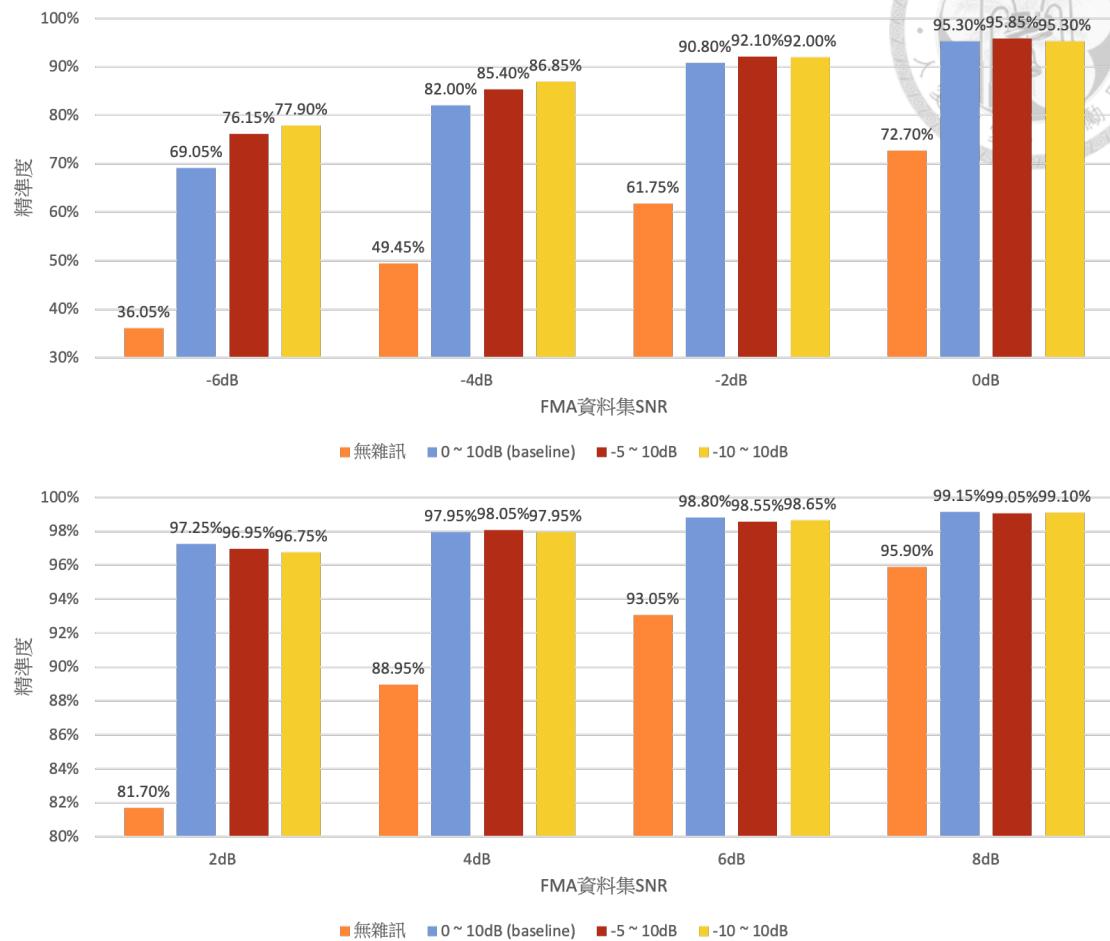
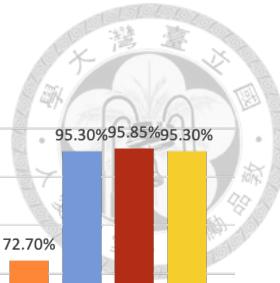


圖 5.6: 雜訊強度實驗在 FMA 驗證集 1 上的精準度

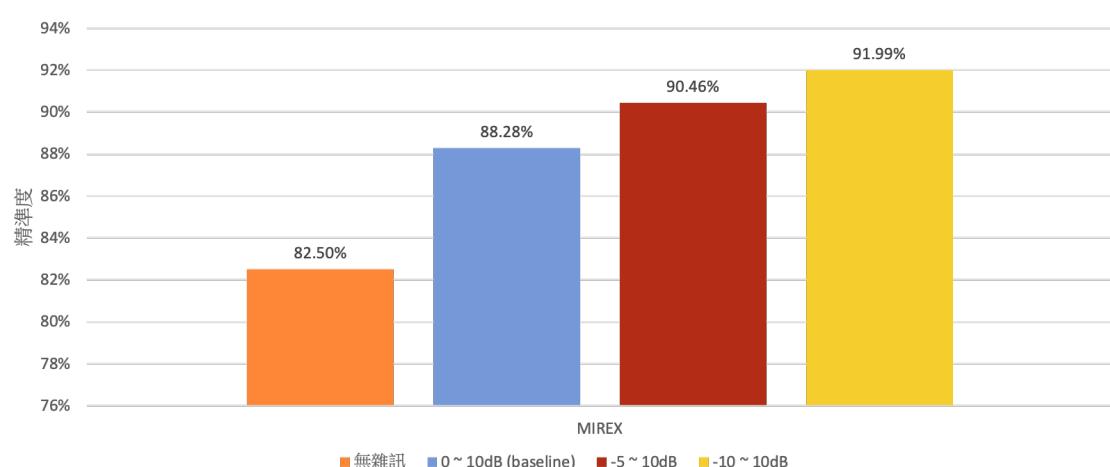


圖 5.7: 雜訊強度實驗在 MIREX 資料集上的精準度



5.5 實驗四：改良資料擴增-殘響

殘響的實驗結果如圖 5.8、圖 5.9、表 5.10 及表 5.11。本實驗比較三種殘響資料擴增的方法，第一種方法是完全沒有殘響，第二種方法是一秒片段的殘響，為 baseline 採用的方法，第三種方法是將一秒片段的前一秒也加入計算殘響，是本研究提出的改進方法。

圖 5.8 是 FMA 資料集上的實驗結果，可以發現，沒有殘響時的精準度最低，而改進方法相較於 Baseline，除了 SNR 是 -6dB 時精準度較高以外，其他的雜訊強度下精準度都和 Baseline 持平，或者更差。

圖 5.9 是 MIREX 資料及上的實驗結果，改進方法的精準度較 Baseline 高出 0.4%，和在 FMA 資料集上的結果不相同。我們認為改進方法在 FMA 上效果較差是因為上一個實驗設定資料擴增雜訊 SNR 可低達 -10dB，對訓練資料造成嚴重干擾，再加上此改進方法後使神經網路更難訓練。

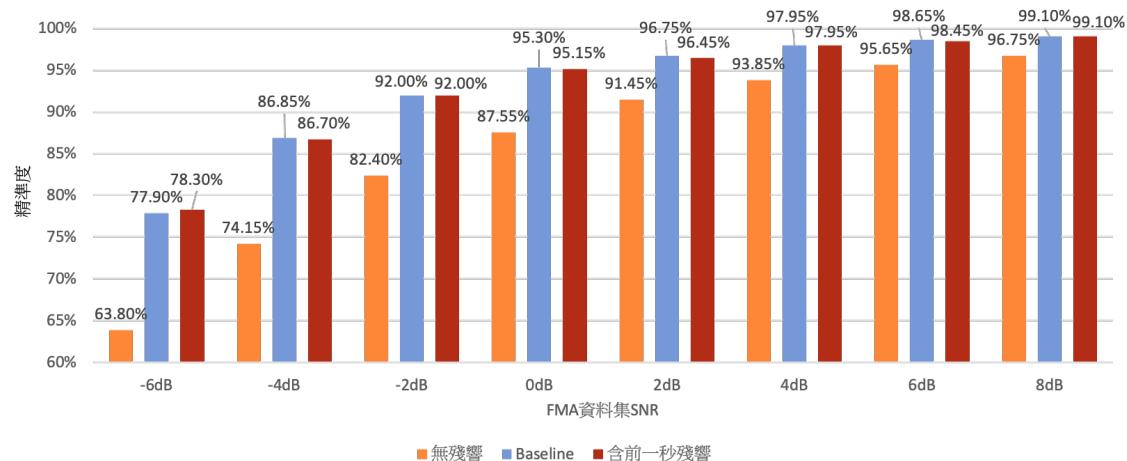


圖 5.8: 殘響實驗在 FMA 驗證集 1 上的精準度

殘響計算範圍	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
無殘響	63.80	74.15	82.40	87.55	91.45	93.85	95.65	96.75	83.63
Baseline	77.90	86.85	92.00	95.30	96.75	97.95	98.65	99.10	91.99
含前 1 秒殘響	78.30	86.70	92.00	95.15	96.45	97.95	98.45	99.10	92.39

表 5.10: 改良殘響資料擴增實驗精準度 (單位 : %)

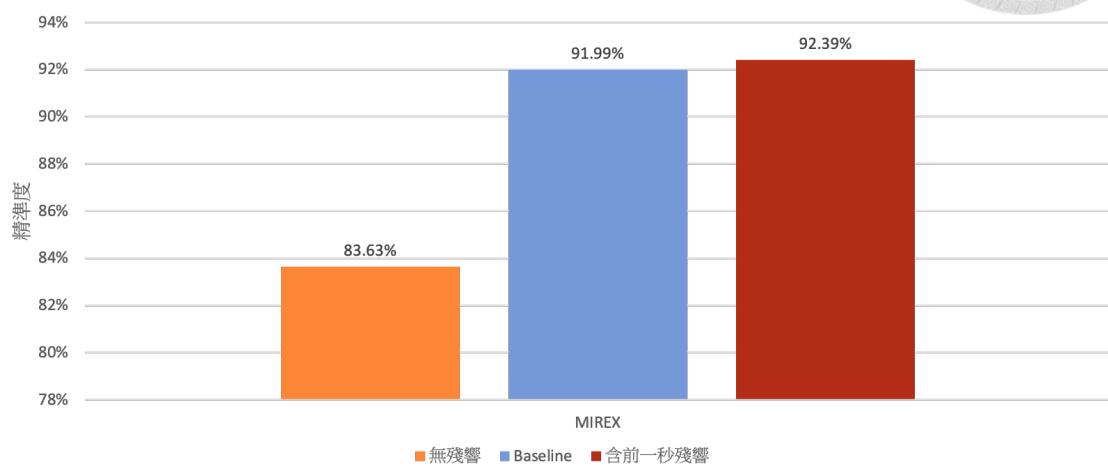


圖 5.9: 殘響實驗在 MIREX 上的精準度

殘響計算範圍	FMA 驗證集 2 雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
無殘響	63.10	73.00	81.75	86.85	91.05	93.55	95.10	96.50
Baseline	78.30	87.20	91.70	94.35	96.45	97.65	98.20	98.75
含前 1 秒殘響	78.95	87.50	92.35	94.65	96.40	97.65	98.20	98.65

表 5.11: 改良殘響資料擴增實驗精準度 (續) (單位 : %)



5.6 實驗五：對查詢做多次時間位移

查詢精準度的結果如圖 5.10、圖 5.11、表 5.12 以及表 5.13， N 是時間位移重複查詢的次數。地標法的 baseline 是 $N = 4$ ，而神經網路法的 baseline 是 $N = 1$ 。在地標法中，增加重複查詢的次數 N 可以有效提升精準度，這與前人的研究結果 [14] 一致，不過相對於 $N = 1$ 到 $N = 2$ 的改進， $N = 2$ 到 $N = 4$ 的改進明顯較小，因為 $N = 1$ 到 $N = 2$ 已經把大部分不精準的配對時間解決， $N = 2$ 到 $N = 4$ 能夠改善的就相對比較少。在神經網路法中，也觀察到一樣的情況，精準度隨著 N 的增大而提升，然而在 FMA 資料集，雜訊 SNR 為 8dB 的情況下， $N = 1$ 的精準度反而比 $N = 2$ 高，可能是因為原本的精準度就已經很高 (99% 以上)，沒有什麼改進空間了。比較地標法和神經網路法則發現，神經網路法的精準度已能超越地標法。

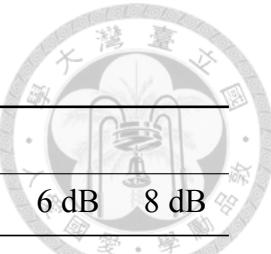
方法	N	FMA 驗證集 1 雜訊 SNR								MIREX
		-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
神經網路法	1	78.30	86.70	92.00	95.15	96.45	97.95	98.45	99.10	92.39
	2	79.85	88.55	93.40	95.95	97.25	98.20	98.60	99.05	93.04
	4	80.35	89.00	93.55	96.10	97.75	98.35	98.75	99.20	93.18
地標法	1	59.70	69.35	77.25	83.95	88.65	91.75	94.20	95.55	83.77
	2	68.40	77.95	84.80	89.80	92.95	94.60	96.20	97.25	89.13
	4	71.10	79.65	86.85	91.10	93.30	95.20	96.60	97.70	89.92

表 5.12: 對查詢做多次時間位移實驗精準度 (單位 : %)

另外來觀察平均一次查詢所需的檢索時間，實驗結果如圖 5.12、圖 5.13 和表 5.14，可以發現到檢索時間隨著 N 的增加而變長，而且神經網路法的速度遠比地標法還慢。

神經網路法在 MIREX 資料集上的檢索時間是在 FMA 資料集的 2.4~2.7 倍，因此神經網路法的檢索時間主要是檢索資料庫的時間，而不是抽取特徵的時間。而地標法在 MIREX 資料集上的檢索時間是在 FMA 資料集的 1.3~1.4 倍，因此地標法的檢索時間主要是抽取特徵的時間。

FMA 資料集在不同的雜訊 SNR 情況下的檢索時間都差不多，表示雜訊強度



方法	N	FMA 驗證集 2 雜訊 SNR							
		-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
神經網路法	1	78.95	87.50	92.35	94.65	96.40	97.65	98.20	98.65
	2	81.15	89.25	93.10	95.35	97.05	98.10	98.50	98.85
	4	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90
地標法	1	57.30	68.45	76.90	83.55	87.40	90.20	92.55	94.05
	2	67.15	77.15	83.85	88.05	91.45	94.25	95.40	96.45
	4	70.65	79.65	85.40	89.80	92.25	94.35	95.55	96.95

表 5.13: 對查詢做多次時間位移實驗精準度 (續) (單位 : %)

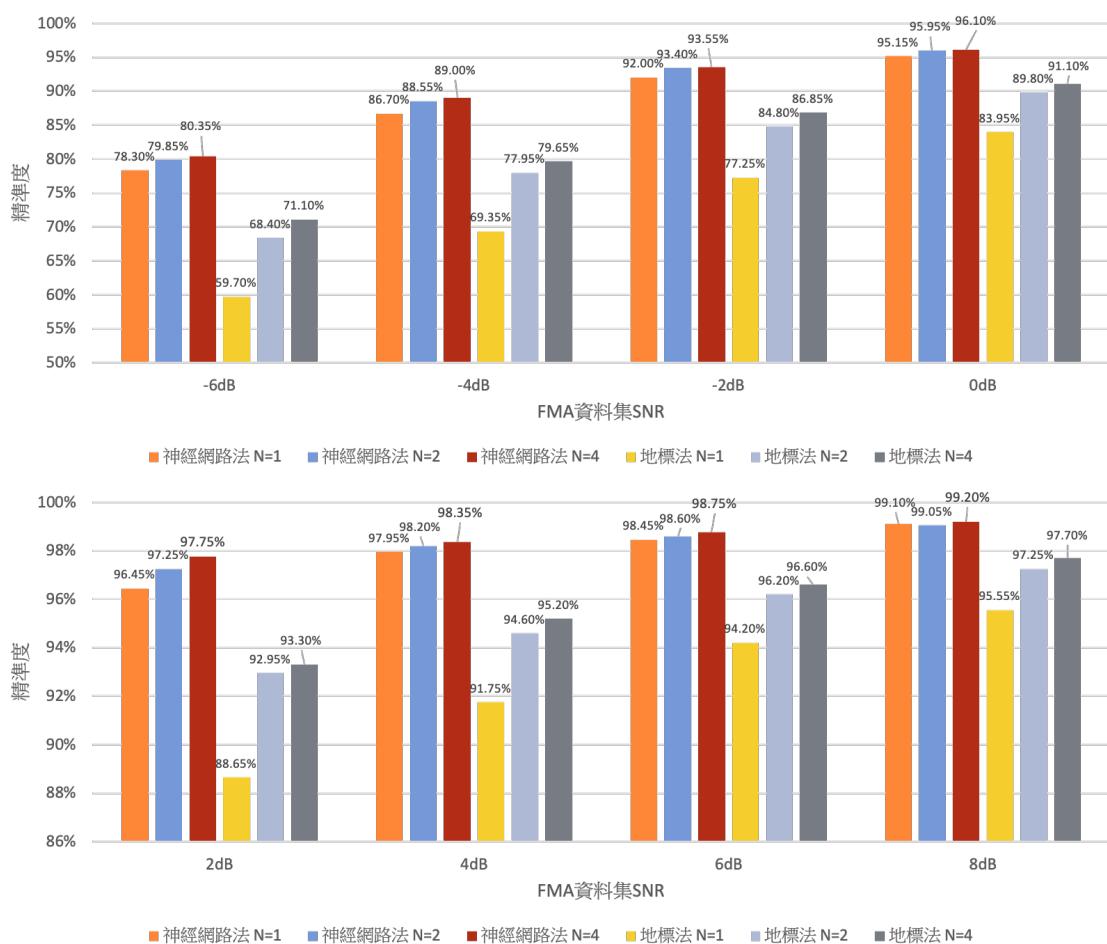


圖 5.10: 對查詢做多次時間位移實驗在 FMA 驗證集 1 上的精準度

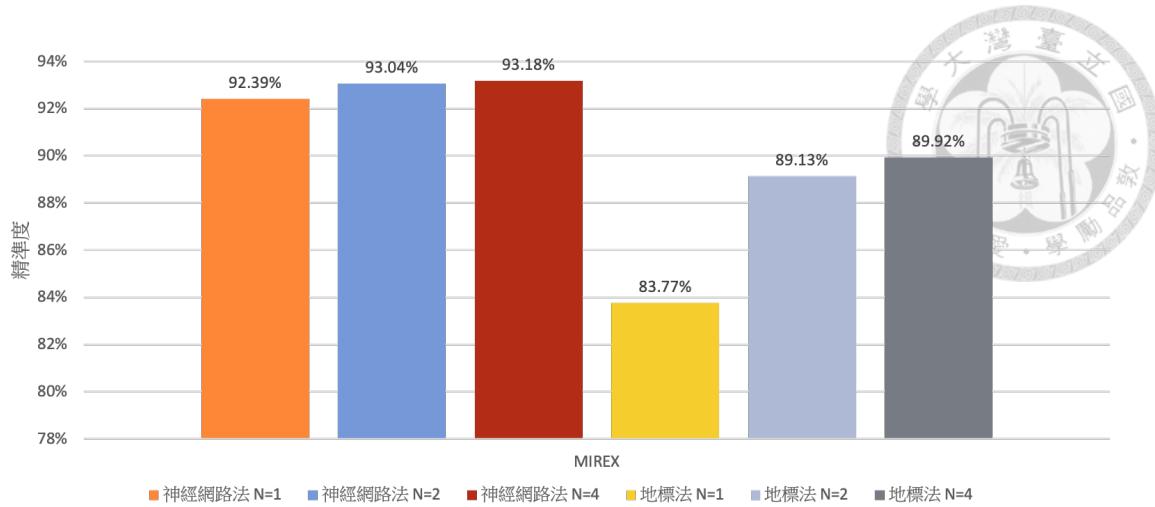


圖 5.11: 對查詢做多次時間位移實驗在 MIREX 資料集上的精準度

不會影響查詢速度。

方法	N	FMA 驗證集 1 雜訊 SNR								MIREX
		-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
神經網路法	1	0.1191	0.1195	0.1174	0.1161	0.1166	0.1197	0.1182	0.1264	0.3178
	2	0.2133	0.2135	0.2131	0.2103	0.2092	0.2102	0.2081	0.2062	0.5546
	4	0.4025	0.3965	0.4242	0.3884	0.3827	0.4112	0.3987	0.3823	1.0147
地標法	1	0.0072	0.0073	0.0073	0.0073	0.0073	0.0075	0.0074	0.0073	0.0095
	2	0.0118	0.0123	0.0118	0.0119	0.0118	0.0118	0.0118	0.0119	0.0160
	4	0.0217	0.0214	0.0214	0.0216	0.0216	0.0216	0.0216	0.0215	0.0290

表 5.14: 對查詢做多次時間位移的平均檢索時間 (單位：秒)

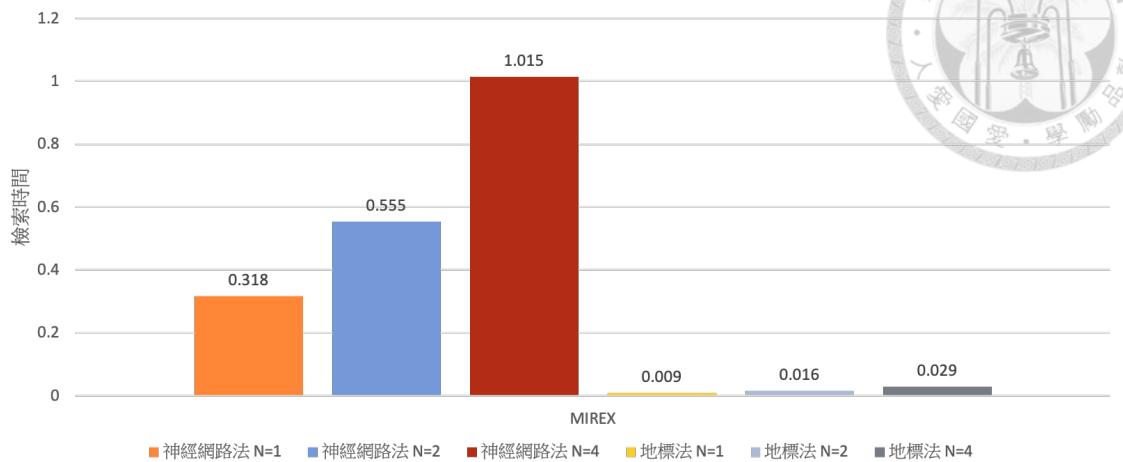


圖 5.12: 對查詢做多次時間位移實驗在 MIREX 資料集上的檢索時間 (單位：秒)

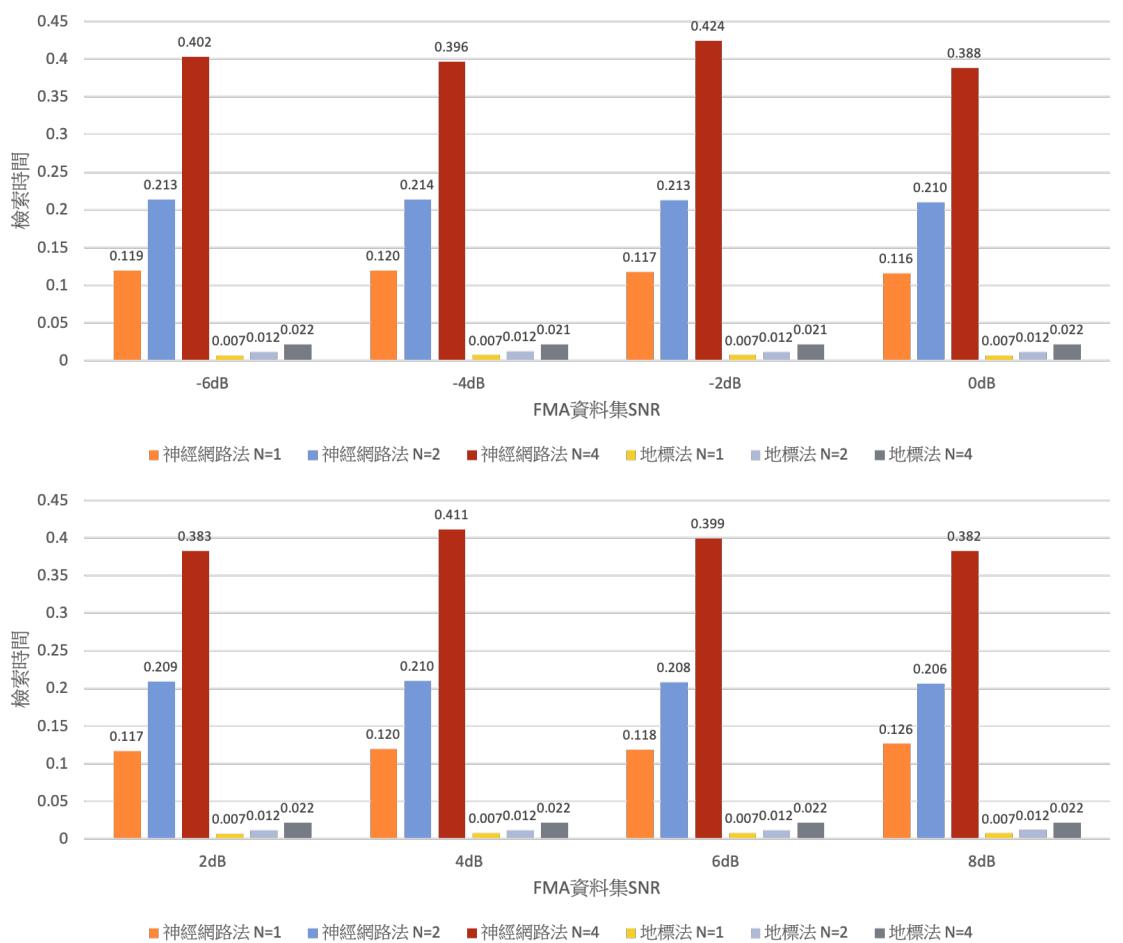


圖 5.13: 對查詢做多次時間位移實驗在 FMA 驗證集 1 上的檢索時間 (單位：秒)



5.7 實驗六：以 SVM 整合地標法和神經網路法的結果

由於 SVM 也需要訓練，因此需要和先前實驗不同的訓練和測試資料。訓練資料根據第 4.6 節的方式來產生，共有 196 個訓練 SVM 的查詢音檔，其中 146 個是只有神經網路法正確的查詢，50 個是只有地標法正確的查詢。

在 FMA 資料集驗證集 2 上，恰有一種方法正確的查詢個數如表 5.15 所示，而在 MIREX 資料集上，有 265 個查詢只有神經網路法正確，79 個查詢只有地標法正確。

雜訊 SNR	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
只有神經網路法正確	300	241	184	132	112	84	67	45
只有地標法正確	79	43	26	16	11	9	6	6
總計	379	284	210	148	123	93	73	51

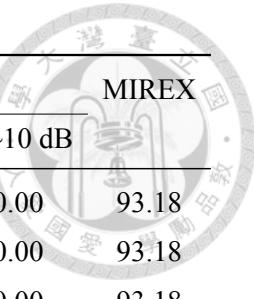
表 5.15: FMA 資料集驗證集 2 上，恰有一種方法正確的查詢個數

5.7.1 線性 SVM

線性 SVM 有一個參數 C ，本研究採用 $C = \{10^{-2}, 10^{-1}, \dots, 10^{10}\}$ 進行測試。實驗的精準度如表 5.16 以及圖 5.14。表 5.16 是考慮所有查詢音檔的精準度，而圖 5.14 只考慮恰有一個方法得到正確答案的查詢的精準度。參數的選擇是根據 FMA 資料集在 SNR 為 -10 到 10dB 的精準度，如表 5.16，在 $C = 10^7$ 時有最高的精準度 91.00%。

從測試結果可以看出， $C = 10^7$ 雖然在 FMA 資料集 SNR 為 -10 到 10dB 的精準度最高，但是在 MIREX 資料集的精準度是 93.94%，並非在 MIREX 資料集的最高精準度 93.97%。同時， $C = 10^{-2}$ 到 10^1 的精準度完全相同，因為此時 SVM 模型在訓練資料的範圍內的輸出皆為「採用神經網路」。

將 $C = 10^7$ 的 SVM 與查詢分數的分布畫在同一張圖上以觀察 SVM，圖 5.15 是 FMA 資料集 SNR 為 -10~10dB 的分數分布，而圖 5.16 是 MIREX 資料集的分數分布，其中橘色點是只有神經網路法正確的查詢，藍色點是只有地標法正確的查詢，黃色區域是 SVM 分類為「神經網路法」的區域，紫色區域是 SVM 分類為



C	FMA 驗證集 2 雜訊 SNR										MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	-10~10 dB		
10^{-2}	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90	90.00	93.18	
10^{-1}	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90	90.00	93.18	
10^0	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90	90.00	93.18	
10^1	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90	90.00	93.18	
10^2	82.10	89.65	93.45	95.65	97.20	98.05	98.60	98.95	90.65	93.59	
10^3	82.35	89.75	93.55	95.75	97.35	98.05	98.60	99.00	90.65	93.73	
10^4	83.10	89.85	93.75	95.85	97.50	98.40	98.75	99.05	90.85	93.87	
10^5	83.50	89.95	93.85	95.85	97.55	98.35	98.75	99.00	90.85	93.92	
10^6	83.45	89.95	93.75	95.85	97.50	98.35	98.75	99.00	90.95	93.96	
10^7	83.45	89.90	93.80	95.90	97.50	98.35	98.75	99.05	91.00	93.94	
10^8	83.45	90.00	93.75	95.85	97.50	98.35	98.75	99.00	90.90	93.97	
10^9	83.25	89.80	93.70	95.75	97.55	98.30	98.75	99.00	90.95	93.96	
10^{10}	83.60	90.05	93.80	95.90	97.50	98.35	98.75	99.05	90.85	93.94	

表 5.16: 使用線性 SVM 的整體查詢精準度和 C 參數的關係 (單位 : %)

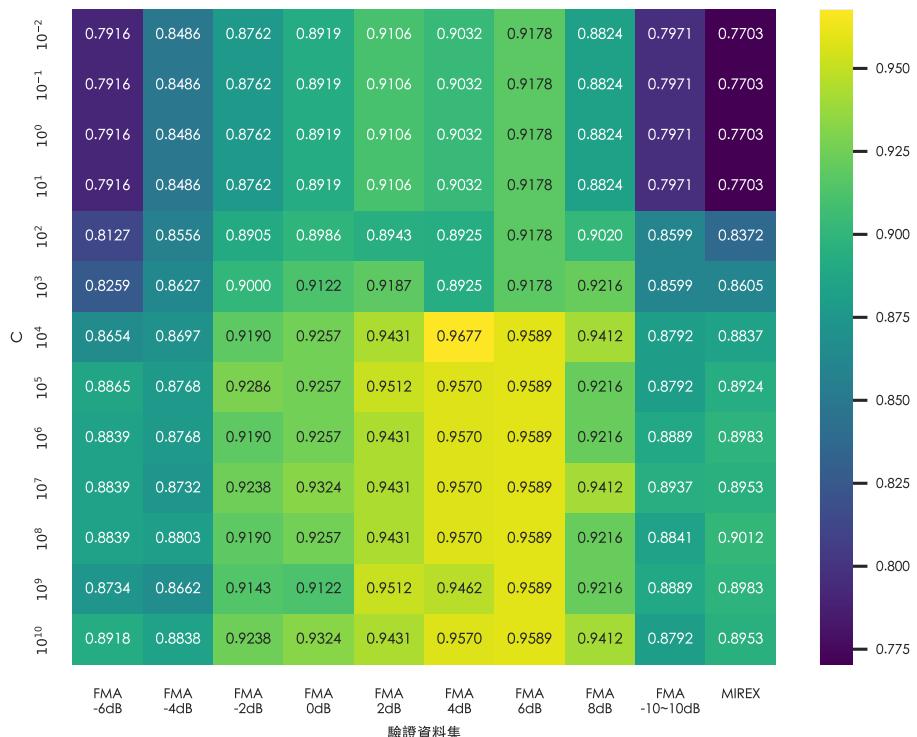


圖 5.14: 使用線性 SVM 的查詢精準度和 C 參數的關係，只考慮恰有一種方法正確的查詢



「地標法」的區域。

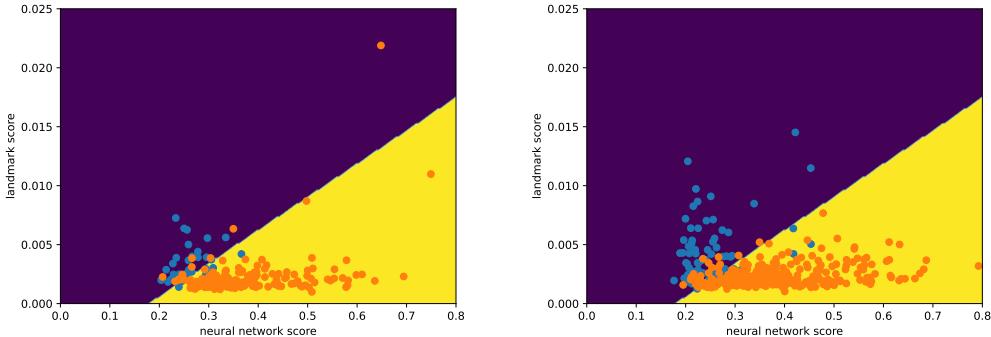


圖 5.15: FMA 資料集 SNR 為 圖 5.16: MIREX 資料集的分數分布以 -10~10dB 的分數分布以及線性 SVM 及線性 SVM 的分界線的分界線

5.7.2 RBF 核 SVM

RBF SVM 有兩個參數 C 和 $gamma$ ，參考 scikit-learn 的範例¹，本研究分別採用 $C = \{10^{-2}, 10^{-1}, \dots, 10^{10}\}$ 和 $gamma = \{10^{-9}, 10^{-8}, \dots, 10^3\}$ 進行測試。圖 5.17 是在 FMA 資料集上測試 SNR 為 -10~10dB 的精準度的熱點圖，測試的音樂是驗證集 2。圖 5.18 是在 MIREX 資料集上測試精準度的熱點圖。參數的選擇是根據 FMA 資料集在 SNR 為 -10 到 10dB 的精準度，如圖 5.17，在 $C = 10^9$ 、 $gamma = 10^{-1}$ 時有最高的精準度 91.10%。然而此參數組合在 MIREX 資料集上的精準度是 93.94%，並非在 MIREX 資料集的最高精準度 93.97%。93.94% 的精準度也和線性 SVM 相同而沒有改進。

將 $C = 10^9$ 、 $gamma = 10^{-1}$ 的 SVM 與查詢分數的分布畫在同一張圖上以觀察 SVM，圖 5.19 是 FMA 資料集 SNR 為 -10~10dB 的分數分布，而圖 5.20 是 MIREX 資料集的分數分布，圖例同上一小節。

¹https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

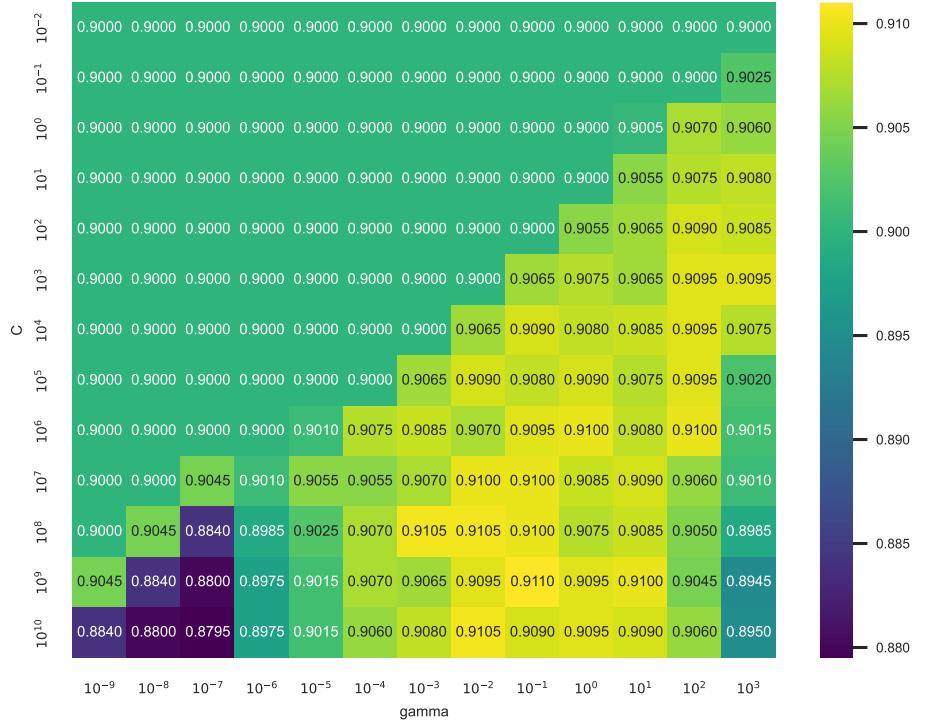


圖 5.17: 使用 RBF 核 SVM 在 FMA 資料集 SNR 為 $-10\sim10$ dB 的整體查詢精準度

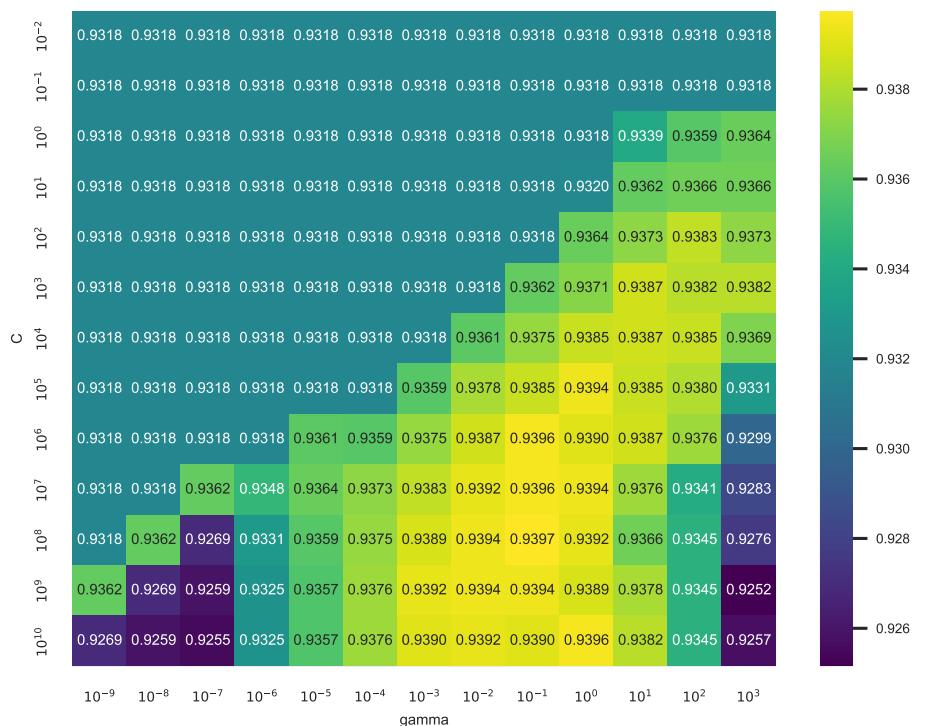


圖 5.18: 使用 RBF 核 SVM 在 MIREX 資料集的整體查詢精準度

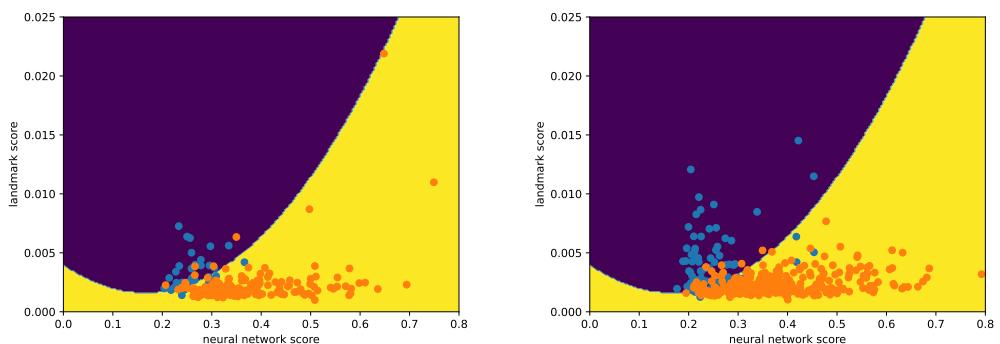


圖 5.19: FMA 資料集 SNR 為 圖 5.20: MIREX 資料集的分數分布以
-10~10dB 的分數分布以及 RBF 核 及 RBF 核 SVM 的分界線
SVM 的分界線

第六章 結論與未來展望



6.1 結論

表 6.1 和表 6.2 是上一章所有實驗中選出的最佳模型，在不同的測試資料上的表現。由上一章的所有實驗，可以歸納出以下結論：

1. 由第 5.1 節的實驗可知，神經網路法在雜訊較弱的情況下精準度較高，而地標法在雜訊較強的情況下精準度較高，並且由於神經網路法的輸出為內嵌向量，比地標法的雜湊還要容易壓縮，因此神經網路法的資料庫大小比地標法小。
2. 由第 5.2 節的實驗一可知，二階段洗牌可以增進模型的精準度，但是 *shuffle_size* 參數需要適中，過大就等同於沒有二階段洗牌，過小則會導致模型難以訓練。
3. 由第 5.3 節的實驗二可知，將資料擴增步驟中的時間位移範圍從 0.2 秒增加到 0.25 秒可以稍微地增加精準度，然而此做法對 MIREX 資料集幾乎沒有效果。
4. 由第 5.4 節的實驗三可知，將資料擴增步驟中的雜訊 SNR 的下限降低可以顯著地增加 FMA 資料集於雜訊較強的情況的精準度，以及 MIREX 資料集的精準度，但是會使雜訊較弱的查詢精準度下降。
5. 由第 5.5 節的實驗四可知，將資料擴增步驟中的殘響加入前一秒的音訊一起計算殘響可以些微的增加 MIREX 資料集精準度，但是卻會些微的降低 FMA 資料集的精準度。
6. 由第 5.6 節的實驗五可知，對查詢做多次時間位移可以增加地標法和神經網路法的精準度，但是會使查詢時間更長。並且發現，神經網路法所需的查詢時間遠比地標法還要長。
7. 由第 5.7 節的實驗六可知，以 SVM 整合地標法和神經網路法的結果可以提升整體的精準度，而線性 SVM 和 RBF 核 SVM 的表現並沒有顯著差異。



改良實驗	FMA 資料集雜訊 SNR							
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB
Baseline (神經網路法)	59.05	75.20	86.40	92.55	95.95	97.30	98.05	99.00
Baseline (地標法)	71.10	79.65	86.85	91.10	93.30	95.20	96.60	97.70
1. 二階段洗牌	66.95	81.50	89.75	95.15	96.90	98.10	98.60	99.15
2. 時間位移	69.05	82.00	90.80	95.30	97.25	97.95	98.80	99.15
3. 雜訊強度	77.90	86.85	92.00	95.30	96.75	97.95	98.65	99.10
4. 殘響	78.30	86.70	92.00	95.15	96.45	97.95	98.45	99.10
5. 對查詢做多次時間位移	80.35	89.00	93.55	96.10	97.75	98.35	98.75	99.20
6.1. 線性 SVM	83.25	90.15	94.25	96.45	97.85	98.40	98.75	99.20
6.2. RBF 核 SVM	83.50	90.55	94.25	96.45	97.85	98.35	98.75	99.25

表 6.1: 以 FMA 驗證集 1 評估各種改良實驗的精準度 (單位 : %)

改良實驗	FMA 資料集雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
Baseline (神經網路法)	58.10	74.30	86.50	92.40	95.30	96.90	97.90	98.40	85.63
Baseline (地標法)	70.65	79.65	85.40	89.80	92.25	94.35	95.55	96.95	89.92
1. 二階段洗牌	65.55	81.15	89.95	94.40	96.35	97.70	98.45	98.85	88.25
2. 時間位移	67.85	81.90	90.05	94.30	96.25	97.70	98.25	98.85	88.28
3. 雜訊強度	78.30	87.20	91.70	94.35	96.45	97.65	98.20	98.75	91.99
4. 殘響	78.95	87.50	92.35	94.65	96.40	97.65	98.20	98.65	92.39
5. 對查詢做多次時間位移	81.70	89.55	93.30	95.60	97.30	98.10	98.60	98.90	93.18
6.1. 線性 SVM	83.45	89.90	93.80	95.90	97.50	98.35	98.75	99.00	93.94
6.2. RBF 核 SVM	83.75	90.30	93.85	96.05	97.55	98.40	98.80	99.05	93.94

表 6.2: 以 FMA 驗證集 2 和 MIREX 資料集評估各種改良實驗的精準度 (單位 : %)



6.2 未來展望

本論文提出的改進使神經網路法的精準度能夠超越地標法的表現，然而還是有一些值得繼續研究的方向。

1. 本論文為了能和現有方法比較，採用了和 NAF 論文 [6] 相同的 FMA 音樂資料集 [8]、AudioSet [11] 的地鐵聲音子集、MicIRP [24] 和 Aachen 資料集 [15]，這些資料集可能和現實音訊指紋檢索的環境有偏差，導致訓練出來的模型表現不佳，或者生成的測試音檔無法忠實呈現出現實的檢索環境。比如 FMA 資料集主要是電子音樂和實驗音樂，然而在生活周遭的音樂主要是流行音樂，在曲風的分布上就出現偏差。AudioSet 包含大量的環境聲音，但本論文使用的地鐵聲音子集可能不足以代表各種環境雜訊，因為查詢有可能遭遇到車輛的噪音、人群喧譁聲、風聲、或者各種機器的噪音等。期望未來能夠修正資料集，以得到更為真實的精準度評估。
2. 在本研究中，我們並未修改 NAF [6] 的模型架構，但是目前已有多種能夠輸入時頻譜的神經網路模型，如 VGG、ResNet 等，這些模型可能有更高的精準度，或著運算速度更快。因此未來也許可以朝著更改模型的方向來研究。
3. 雖然神經網路法的精準度比地標法更高，但是即使都有 CPU 平行化的情況下，神經網路法的檢索速度還是遠比地標法慢。以 MIREX 資料集來看，神經網路法的查詢時間是 1.0147 秒，而地標法是 0.0290 秒，差距將近 35 倍，因此要大規模使用神經網路法之前，需要解決檢索速度的問題。可能的做法有改進內嵌向量資料庫的索引方式，因為大部分的檢索時間都是在搜尋最相似的內嵌向量。或者使用 GPU 加速，因為 NAF 論文 [6] 宣稱使用 GPU 檢索 100,000 首音樂的資料庫只需要 0.02 秒。
4. 本論文中的 SVM 模型用來判斷地標法和神經網路法，何者較可能得到正確答案，但是輸入特徵為兩種方法的配對分數，以至於檢索需要兩種方法皆完成才能回傳答案。如果輸入特徵改為不需要神經網路法的特徵，有可能 SVM 可以決定跳過神經網路法，只用地標法來檢索，如此也能夠加速檢索。

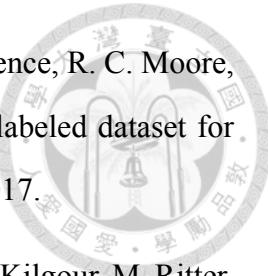
未來也許可以研究在地標法和神經網路法之前利用 SVM 決定適合的音訊指紋演算法。



參考文獻



- [1] 唐子翔。「以雙向檢索及排序學習演算法來改進音訊指紋辨識」。碩士論文，國立臺灣大學資訊工程學研究所，2020。
- [2] 廖信富。「藉由目標區域以及雜湊表調整對以地標為特徵音訊指紋的改進」。碩士論文，國立臺灣大學資訊工程學研究所，2018。
- [3] 廖珮妤。「用於音樂檢索的聲紋辨識改良」。碩士論文，國立清華大學資訊工程學系，2013。
- [4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint*, 2016. arXiv:1607.06450.
- [5] A. Báez-Suárez, N. Shah, J. A. Nolazco-Flores, S.-H. S. Huang, O. Gnawali, and W. Shi. Samaf: Sequence-to-sequence autoencoder model for audio fingerprinting. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(2), May 2020.
- [6] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han. Neural audio fingerprint for high-specific audio retrieval based on contrastive learning. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*, 2021.
- [7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.
- [8] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson. Fma: A dataset for music analysis. *arXiv preprint*, 2017. arXiv:1612.01840.
- [9] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint*, 2017. arXiv:1708.04552.
- [10] W. Drevo. Dejavu: Audio fingerprinting and recognition algorithm implemented in python. <https://github.com/worldveil/dejavu>, 2014.



- [11] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.
- [12] B. Gfeller, B. Aguera-Arcas, D. Roblek, J. D. Lyon, J. J. Odell, K. Kilgour, M. Ritter, M. Sharifi, M. Velimirović, R. Guo, and S. Kumar. Now playing: Continuous low-power music recognition. In *NIPS 2017 Workshop: Machine Learning on the Phone*, 2017.
- [13] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *ISMIR*, 2002.
- [14] R. Jang. 16-1 landmark extraction. <http://mirlab.org/jang/books/audioSignalProcessing/afpLandmarkExtraction.asp?title=16-1%20Landmark%20Extraction>.
- [15] M. Jeub, M. Schäfer, and P. Vary. A binaural room impulse response database for the evaluation of dereverberation algorithms. In *Proceedings of International Conference on Digital Signal Processing (DSP)*, pages 1–4, Santorini, Greece, July 2009. IEEE, IET, EURASIP, IEEE.
- [16] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.
- [17] MIREX. 2020:audio fingerprinting. https://www.music-ir.org/mirex/wiki/2020:Audio_Fingerprinting. Accessed: 2021-4-23.
- [18] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.



- [20] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.
- [21] Shazam. <https://www.shazam.com>.
- [22] Soundhound. <http://www.soundhound.com/>.
- [23] A. Wang. An industrial strength audio search algorithm. In *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, 2003.
- [24] Xaudia. Microphone impulse response project. <http://micirp.blogspot.com>, 2017.

附錄 A — 神經網路法效能分析

表 A.3 為實驗最終模型的神經網路法在搜尋時，各個步驟的平均執行時間，使用的測試資料為 FMA 資料集的驗證集 1、以及 MIREX 資料集。表 A.4 則是嘗試以 GPU 加速神經網路法中的 KNN 搜尋步驟，所得到的執行時間，使用的測試資料和表 A.3 相同。

步驟	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
讀檔	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	11.8
重新取樣	0.7	0.7	0.8	0.7	0.7	0.8	0.7	0.7	4.8
轉為單聲道	0.3	0.3	0.4	0.3	0.3	0.3	0.3	0.3	0.3
計算內嵌向量	51.4	51.3	51.2	51.3	51.3	51.3	51.3	51.3	51.2
KNN 搜尋	99.9	99.7	98.3	98.6	96.9	100.1	96.5	97.4	678.6
重新計分	248.6	242.8	272.0	236.0	232.0	257.1	248.3	231.0	266.9
其他	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
總計	402.5	396.5	424.2	388.4	382.7	411.2	398.7	382.3	1014.7

表 A.3: 神經網路法在搜尋時各個步驟的平均執行時間 (單位：毫秒)

步驟	FMA 驗證集 1 雜訊 SNR								MIREX
	-6 dB	-4 dB	-2 dB	0 dB	2 dB	4 dB	6 dB	8 dB	
讀檔	1.5	9.2	1.5	1.5	1.5	1.4	1.4	1.5	34.3
重新取樣	0.7	0.8	0.7	0.7	0.7	0.7	0.7	0.7	4.7
轉為單聲道	0.3	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3
計算內嵌向量	51.0	51.3	51.3	51.3	51.0	51.3	51.2	51.1	51.2
KNN 搜尋	3.8	3.8	3.7	3.7	3.7	3.7	3.7	3.7	18.7
重新計分	244.4	251.9	239.6	238.8	235.5	229.8	228.3	241.6	269.6
其他	0.9	0.9	0.9	0.8	0.8	0.8	0.8	0.8	1.1
總計	302.5	318.2	298.0	297.1	293.5	288.1	286.5	299.7	380.0

表 A.4: 神經網路法以 GPU 加速 KNN 搜尋，在搜尋時各個步驟的平均花費時間 (單位：毫秒)