creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사학위논문

아티스트 정보 활용을 통한 음악 장르 분류 성능 향상

Improving the Performance of Music Genre Classification through Exploiting Artist Information

2023 년 2 월

서울대학교 대학원

산업공학과

이 석 기

아티스트 정보 활용을 통한 음악 장르 분류 성능 향상

Improving the Performance of Music Genre Classification through Exploiting Artist Information

지도교수 이 경 식

이 논문을 공학석사 학위논문으로 제출함

2023 년 1 월

서울대학교 대학원

산업공학과

이 석 기

이석기의 공학석사 학위논문을 인준함

2023 년 1 월

위 원 장 조 성 준 (인)

부위원장 이 경 식 (인)

위 원 박 종 헌 (인)

초록

음악 장르 분류는 대상 음악을 블루스, 재즈, 클래식, 팝 등과 같은 특정 장르로 분류하는 과업이다. 음악 장르 분류에서는 장르별로 음악적 특성이 다르다는 점을 이용하여 장르별 특성을 파악하고 추출하는 것이 중요하다. 따라서 특징 추출과 분류 과정이 하나의 신경망 네트워크 구조를 통해 해결되는 딥러닝 모델이 최근 많은 관심을 끌고 있다. 본 연구에서는 딥러닝 모델을 활용하여 음악 장르 분류를 한다.

대다수의 딥러닝 모델들은 곡만의 정보만을 활용하여 음악 장르 분류를 시도한다. 하지만 많은 사람은 곡의 아티스트만을 통해 곡의 장르를 예측하기도 하며 이는 높은 정확도를 보인다. 본 논문에서는 이러한 점에 착안하여 딥러닝 모델을 통해 곡의 정보에 사이드 정보인 아티스트 정보를 추가 활용해 음악 장르 분류에 활용하려고 한다. 아티스트 정보가 장르 구분에 주요한 역할을 한다는 것을 아티스트 정보만을 활용한 장르 분류 실험 결과를 통해 확인하고 아티스트 정보를 효과적으로 활용할 수 있는 그래프 뉴럴 네트워크 기반 플러그 앤 플레이 프레임워크를 제안한다. 아티스트 정보를 아티스트 임베딩을 통해 직접적으로 제공하는 방법과 그래프 뉴럴 네트워크를 통해 간접적으로 제공하는 방법을 비교 실험하고 아티스트 정보를 직접적으로 제공하는 방법의 한계점과 그에 대비되는 그래프 뉴럴 네트워크의 장점을 제시한다.

음악 장르 분류 연구에서는 다양한 데이터 형태의 음악을 활용하여 음악적 특성을 딥러닝 모델에 학습한다. 음악 데이터 형태로는 크게 오디오 데이터를 이미지 형태로 변환한 후 데이터로 활용하는 방법과 심볼릭 형태로 변환하여 활용하는 방법 두 가지가 존재한다. 본 연구에서는 해당 프레임워크가 플러그 앤 플레이인 것을 검증하기 위해 두 가지 데이터 형태의 오픈 데이터셋에 대해 각각 제안 기법을 실험한다. 또한, 해

당 데이터셋으로 학습하는 네 가지 representation 모델에 제안 기법을 적용하고 비교 실험한다. 네 가지 모델들은 각기 다른 방법으로 곡 representation을 학습했다. 두 가지 모델은 지도 학습 방식(supervised learning)으로 학습했고 나머지 두 가지 모델은 비지도 학습 방식(unsupervised learning)으로 음악적 특성을 사전 학습 후 지도 학습 방식으로 모델을 음악 장르 분류에 맞게 파인 튜닝(fine-tuning)했다. 다양한 방법으로 학습된 모델을 통해 제안 기법이 다양한 환경에서 적용 가능하다는 것을 확인한다.

주요어: 음악 장르 분류, 아티스트 정보, 그래프 뉴럴 네트워크, 플러그 앤 플레이

학번: 2021-22750

목차

초록 **i**

목차 **iv**

표 목차 **v**

그림 목차 **vi**

제 1 장 서론 **1**

1.1 연구 배경 및 동기 1

1.2 연구 목적 2

1.3 문제 정의 3

1.4 논문구성 6

제 2 장 배경 이론 및 관련 연구 **7**

2.1 Representation 모델 7

2.2 그래프 뉴럴 네트워크 16

2.3 관련 연구 20

제 3 장 아티스트 정보를 효과적으로 활용할 수 있는 그래프 뉴럴 네트워크 기반 플러그 앤 플레이 프레임워크 **23**

3.1 제안 기법 구조 23

3.2 베이스라인 모델 25

3.3 그래프 뉴럴 네트워크 모델 26
3.4 아티스트 정보 활용 방법 28

제 4 장 실험 결과 **29**
4.1 데이터셋 29
4.2 실험 세팅 34
4.3 실험 결과 39
4.3.1 MusicBERT 39
4.3.2 피아노롤 40
4.3.3 Mixture of CNN 41
4.3.4 CoLA 42
4.4 Label homophily ratio 44
4.5 아티스트 정보 사용 방법 비교 45
4.6 장르 개수에 따른 베이스라인 모델과의 성능 차이 및 각 모델별 성능 표준 편차 46

제 5 장 결론 **50**
5.1 결론 50
5.2 한계 51
5.3 향후 연구 51

참고문헌 **53**

Abstract **63**

표 목차

표 4.1 MusicBERT 하이퍼파라미터 35
표 4.2 피아노롤 모델 하이퍼파라미터 36
표 4.3 Mixture of CNN 하이퍼파라미터 37
표 4.4 CoLA 하이퍼파라미터 38
표 4.5 MusicBERT 실험 결과 39
표 4.6 피아노롤 모델 실험 결과 40
표 4.7 Mixture of CNN 실험 결과 41
표 4.8 CoLA 실험 결과 42
표 4.9 각 데이터셋 별 Label Homophily Ratio 44
표 4.10 MusicBERT - 장르 개수에 따른 성능 표준 편차 47
표 4.11 피아노롤 - 장르 개수에 따른 성능 표준 편차 48
표 4.12 CoLA - 장르 개수에 따른 성능 표준 편차 48

그림 목차

그림 2.1 피아노롤 형태[21] 7

그림 2.2 MusicBERT 구조[65] 8

그림 2.3 Bert Masked Language Modeling(MLM)[5] 9

그림 2.4 MusicBERT 마스킹 방법[5] 10

그림 2.5 Mixture of CNN 구조[64] 11

그림 2.6 스펙트로그램[64] 12

그림 2.7 오리지널, 하모닉, 퍼커시브 스펙트로그램[64] 13

그림 2.8 CoLA 구조[52] 14

그림 2.9 그래프 뉴럴 네트워크의 타겟 노드 업데이트 방법[1] 17

그림 2.10 그래프 뉴럴 네트워크 계산 그래프[3] 17

그림 3.1 제안 기법 구조 23

그림 4.1 FMA Small 데이터셋 장르 구성 29

그림 4.2 Lakh MIDI 데이터셋 장르 구성 30

그림 4.3 Lakh Pianoroll 데이터셋 장르 구성 31

그림 4.4 MTG-Jamendo 데이터셋 장르 구성 32

그림 4.5 MusicBERT - 장르 개수 별 베이스라인과 제안 기법 성능 차이 . 46

그림 4.6 피아노롤 - 장르 개수 별 베이스라인과 제안 기법 성능 차이 . . . 47

그림 4.7 CoLA - 장르 개수 별 베이스라인과 제안 기법 성능 차이 48

제 1 장 서론

1.1 연구 배경 및 동기

음악은 리듬이나 화성, 사용 악기와 같은 다양한 특성이 결합하여 한 곡을 이룬다. 여러 특성이 결합하여 표현된 하나의 곡은 형식, 전통, 관습에 따라 특정 장르에 속하게 된다. 결합된 다양한 특성 중 각 장르를 구별할 수 있는 적절한 특성을 찾아낸 후 음악 장르 분류에 활용하는 것이 중요하다. 초기 음악 장르 분류에서는 장르 별로 다른 음악적 특성을 하나하나 찾아낸 후 해당 특성을 추출하고 분류를 진행하였다[18]. 최근에는 특징 추출과 분류 과정이 하나의 신경망 네트워크를 통해 해결되는 딥러닝 모델[10]에 많은 연구가 진행되고 있다.

현재까지의 딥러닝을 활용한 음악 장르 분류 연구에서는 곡만을 활용해 음악 장르 분류를 수행하는 시도가 대다수였다. 하지만 많은 사람은 아티스트만을 활용해 곡의 장르를 예측하기도 하며 이는 높은 정확도를 보인다. 예를 들어, 한국 가수 성시경의 경우 2000년 데뷔 이후 7개의 정규 앨범, 3개의 리메이크 및 스페셜 앨범, 1개의 베스트 앨범을 발매하며 지금까지 꾸준히 가수로서 활동하고 있지만 그의 노래 장르는 대다수 Pop 또는 R&B로 이뤄져 있다. 따라서, 가수인 성시경만을 보고 해당 노래의 장르를 예측하면 매우 높은 정확도를 보일 수 있다. 또한, 최근에는 아티스트 정보와 같은 사이드 정보를 활용해 데이터 간의 관계를 정의하는 연구가 최근 많은 관심을 받고 있고 이러한 관계를 학습할 수 있는 모델인 그래프 뉴럴 네트워크[53] 또한 많은 관심을 받고 있다. 따라서, 본 연구에서는 딥러닝 모델 중 하나인 그래프 뉴럴 네트워크[53]를 통해 곡의 정보에 아티스트 정보를 추가 활용하여 음악의 장르를 분류하는 기법을 제안한다.

1.2 연구 목적

본 연구에서는 아티스트 정보를 데이터로써 곡의 정보와 함께 활용한다. 아티스트 정보가 장르 구분에 주요한 역할을 한다는 것을 네 가지 오픈 데이터셋[49, 21, 16, 9]을 통해 보이고 아티스트 정보를 효과적으로 활용할 수 있는 그래프 뉴럴 네트워크[53] 기반 플러그 앤 플레이 프레임워크를 제안한다. 플러그 앤 플레이임을 검증하기 위해 각자 다른 방식으로 음악 장르 분류 과업을 수행한 네 가지 모델의 뒷부분에 제안 기법인 그래프 뉴럴 네트워크를 추가 적용하여 실험한다. 기존 네 가지 모델들은 곡만의 정보를 사용해 음악의 장르를 분류하였다. 이러한 기존 모델들의 방법을 곡만을 사용한 방법이라고 정의한다.

곡만을 사용할 때, 아티스트 정보만을 사용할 때, 아티스트 정보를 임베딩을 통해 직접적으로 추가 사용할 때, 그래프 뉴럴 네트워크를 활용해 간접적으로 사용할 때에 대해 비교 실험을 진행한다. 실험들을 바탕으로 곡만을 사용했을 때와 아티스트만을 사용했을 때를 비교하여 아티스트 정보가 음악 장르 분류에 얼마큼의 영향을 보이는지 알 수 있다. 또한, 아티스트 정보를 직접적으로 사용한 경우와 간접적으로 사용했을 때를 비교하여 아티스트 정보를 직접적으로 사용했을 때의 한계를 제시하고 그와 대비되는 간접적 사용의 상대적 장점을 제시한다. 그래프 구조를 이용해 간접적으로 아티스트 정보를 활용할 때 두 노드가 같은 아티스트일 경우 두 노드는 서로를 이웃 노드로 정의한다. 이렇게 정의된 이웃 노드끼리 라벨이 일치하는 비율(label homophily ratio)을 네 가지 데이터셋에 대해 각각 계산하고 계산된 값들을 토대로 다양한 환경에서 해당 제안 기법의 성능을 확인한다. 곡의 정보에 아티스트 정보를 추가로 사용할 때 아티스트가 여러 장르를 가진다면 아티스트 정보가 노이즈로 작용할 가능성이 존재하므로 아티스트가 가지는 장르의 수에 따라 곡만을 활용한 모델과 간접적으로 아티스트 정보를 활용한 모델과의 성능 차이를 제시한다.

1.3 문제 정의

음악은 여러 가지 데이터 형태로 표현될 수 있다. 대표적으로 오디오 데이터가 존재한다. 하지만 오디오 데이터는 데이터의 수 대비 크기가 매우 커 제한된 리소스 내에서 많은 데이터를 활용할 수 없는 한계점을 지니고 있다. 이러한 한계를 해결할 수 있는 데이터 형태가 심볼릭 데이터이다. 심볼릭 데이터는 음악을 특정 의미를 담고 있는 숫자나 알파벳으로 표현하는 것이다. 대표적인 예로 MIDI와 피아노롤이 있다. MIDI는 다양한 전자 악기들이 연주 데이터를 전송하고 공유할 수 있도록 정한 디지털 표준 규격이다. 피아노롤은 특정 피치의 존재 여부를 이진값으로 표현한 형태이다. 심볼릭 데이터는 오디오 데이터에 비해 대략 $200 \sim 1000$배 가량 적은 크기를 가진다고 알려져 있다[2]. 하지만 심볼릭 데이터를 활용한 연구는 오디오 데이터를 활용한 연구에 비해 최근까지는 활발히 연구가 진행되지 않았다. 또한, 오디오 데이터를 심볼릭 형태로 표현하면 정보의 손실이 일어날 위험도 존재한다. 하지만, 두 가지의 데이터 형태 모두 하나의 곡을 딥러닝 모델이 학습할 수 있는 벡터의 형태로 표현할 수 있으므로 본 연구에서는 대표적인 음악 데이터 형태인 오디오와 심볼릭 형태에 대해 모두 제안 기법을 실험하고자 한다. 따라서, 오디오 오픈 데이터셋 2가지와 심볼릭 오픈 데이터셋 2가지를 본 연구의 데이터셋으로 사용한다.

딥러닝에서 representation이란 입력 데이터를 모델이 학습한 것을 토대로 해석하여 표현한 정보라고 할 수 있다. 같은 대상이어도 학습하는 방법에 따라 다양하게 표현될 수 있으며 높은 품질의 representation은 대상 과업을 더 수월하게 해결할 수 있게 해준다[27]. 본 연구에서는 곡에 대한 representation을 생성할 수 있게 학습된 모델을 representation 모델이라고 정의하고 곡 representation을 곡 정보라고 정의한다. Representation 모델로는 총 4가지를 사용한다. 4가지 모델 중 두 가지 모델은 심볼릭 데이터를 사용하는 모델이고 나머지 두 가지 모델은 오디오 데이터를 사용하는 모델이다.

각각의 두 가지 모델 중 하나의 모델은 지도 학습(supervised learning)을 통해 모델을 학습하고 나머지 한 가지 모델은 음악 representation을 사전 학습(pre-training)[56]한 후 학습된 representation을 음악 장르 분류에 활용하는 모델이다. 해당 모델은 사전 학습이 필요하므로 비지도 학습(unsupervised learning)을 통해 모델을 사전 학습시킨 후 지도 학습을 통해 음악 장르 분류에 맞게 파인 튜닝한다. 따라서, 네 가지 모델들은 학습 데이터 형태가 다르거나 학습 방법이 다른 모델들로 구성하여 다양한 환경에서의 제안 기법 효과를 확인하려고 한다.

기존 네 가지 representation 모델들은 곡만의 정보를 사용해 음악의 장르를 분류하였으므로 기존 실험의 결과를 곡만을 사용한 베이스라인으로써 활용한다. 또한, 아티스트 정보만을 활용할 때 얼마큼의 성능을 보일 수 있는지를 알기 위해 의사결정나무[48] 모델을 사용하여 음악 장르를 분류한다. 한 가지 정보인 곡 정보만을 사용하였을 때와 두 가지 정보 모두를 사용할 때의 성능 비교를 통해 추가 활용하는 아티스트 정보가 음악 장르 분류에 어떠한 영향을 미치는지 확인한다. 추가 활용하는 아티스트 정보를 직접적으로 임베딩을 통해 모델에 제공하는 경우와 그래프 뉴럴 네트워크를 통해 간접적으로 사용하는 경우를 비교한다.

Representation 모델로는 심볼릭 데이터인 MIDI 데이터를 입력값으로 활용하고 비지도 학습 방법의 하나인 자기 지도 학습(self-supervised learning)으로 음악 정보를 사전 학습 후 학습된 곡 representation을 음악 장르 분류에 맞게 파인 튜닝하여 활용하는 MusicBERT[65], 전통적인 음악 표현 방식인 피아노롤 데이터를 입력값으로 활용하고 지도 학습으로 음악의 장르를 분류하는 피아노롤 모델[21], 오디오 데이터를 입력값으로 활용하고 지도 학습 방법으로 음악의 장르를 분류하는 Mixture of CNN[64], 마찬가지로 오디오 데이터를 입력값으로 활용하면서 비지도 학습 방법의 하나인 대조적 학습(contrastive learning)으로 사전 학습 후 학습된 곡 representation을 음악 장르 분류에

맞게 파인 튜닝하여 활용하는 CoLA[52]를 사용한다. Representation 모델에 대한 보다 상세한 설명은 2.1절에서 서술한다.

1.4 논문구성

본 논문은 다음과 같이 총 5장으로 구성된다. 제 2장에서는 본 연구에서 사용하는 representation 모델들과 그래프 뉴럴 네트워크에 대해 설명하고 관련된 선행 연구를 살펴본다. 제 3장에서는 본 연구에서 사용하는 네 가지 데이터셋에 대해 소개하고 제안 기법의 개요, 제안 기법과 비교할 베이스라인 모델, 제안 기법에서 사용하는 그래프 뉴럴 네트워크 모델, 아티스트 정보 활용 방법에 대해 설명한다. 제 4장에서는 네 가지 representation 모델의 실험 결과에 대해 각각 살펴본다. 마지막으로 제 5장에서는 본 연구의 결론과 한계점에 대해 논의하고 향후 연구방향을 제시한다.

제 2 장 배경 이론 및 관련 연구

2.1 Representation 모델

Representation이란 정보를 어떻게 가공하여 표현할지를 뜻하는 것으로 학습하는 방법에 따라 다양하게 표현될 수 있다. 본 연구에서는 총 네 가지 representation 모델로 피아노롤[21], MusicBERT[65], Mixture of CNN[64], CoLA[52]를 사용한다. 해당 모델들은 다양한 방법으로 음악적 특성(representation)을 학습했다. 이번 절에서는 해당 모델들에 대해 설명한다.

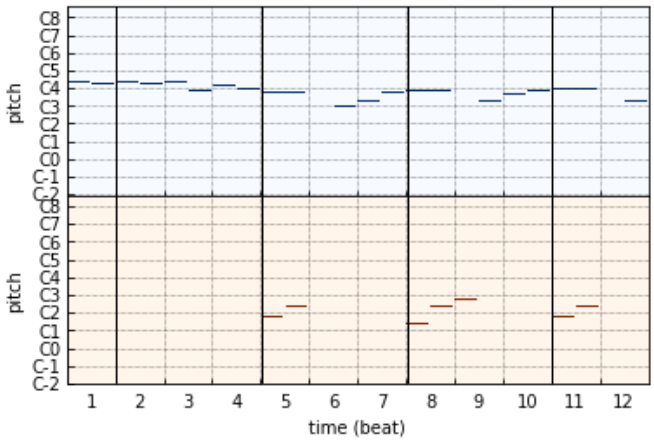

Figure 2.1 피아노롤 형태[21]

첫 번째 모델은 피아노롤 모델이다. 심볼릭 데이터인 MIDI 데이터를 음악에서 전통적으로 자주 사용했던 곡 표현 방식인 피아노롤 형태로 변환시켜 모델의 데이터로 활용한다. 그림 2.1에 피아노롤 형태에 대한 예시가 있다. X축은 시간이고 Y축은 피치를 나타낸다. 피아노롤 표현은 이진수 값으로 특정 시간에서의 특정 피치의 존재 여부를

나타낸다. 초기에는 피아노 트랙에 대해서만 피아노롤 형태로 변환 후 활용하였지만, 현재는 드럼, 베이스, 기타 등 여러 가지 트랙도 피아노롤 형태로 변환 후 행렬 형태로 합쳐 사용한다. 4가지 모델 중 유일하게 모델의 출력이 아닌 데이터 자체를 곡 정보라고 여긴다. 해당 모델의 구조는 6개의 콘볼루션 신경망과 6개의 선형 레이어 이뤄져 있고 지도 학습(supervised learning) 방식으로 학습이 진행된다.

두 번째 모델은 MusicBERT[65]이다. 그림2.2에 모델 구조를 나타냈다. 해당 모델은 심볼릭 데이터인 MIDI 데이터를 활용하며 자연어 처리 분야(natural language processing)의 BERT[17] 모델을 음악 도메인에 적용한 모델이다. 자연어 처리에서의 하나의 단어를 하나의 노트로 매칭시켜 음악에 적용하였다. 하지만 하나의 단어를 하나의 노트로 매칭시킬 때는 단어와는 다른 점을 고려해야 한다. 음악에서는 하나의 노트가 음악의 구조적 정보(e.g., 바에서의 위치, 전체 곡에서의 바의 순서 등)와 다양한 정보(e.g., 템포, 악기, 피치)를 담고 있다. 같은 피치를 가진 노트라도 노트의 위치와 노트를 연주하는 악기에 따라 해당 노트의 의미는 달라질 수 있다. 반면 언어에서는 하나의 단어가 해당 단어의 뜻만을 텍스트 형태로 담고 있다. 단어의 위치가 바뀌면 문법적으로는 문제가 될 수 있지만, 그 단어 자체가 가진 의미는 달라지지 않는다. 따

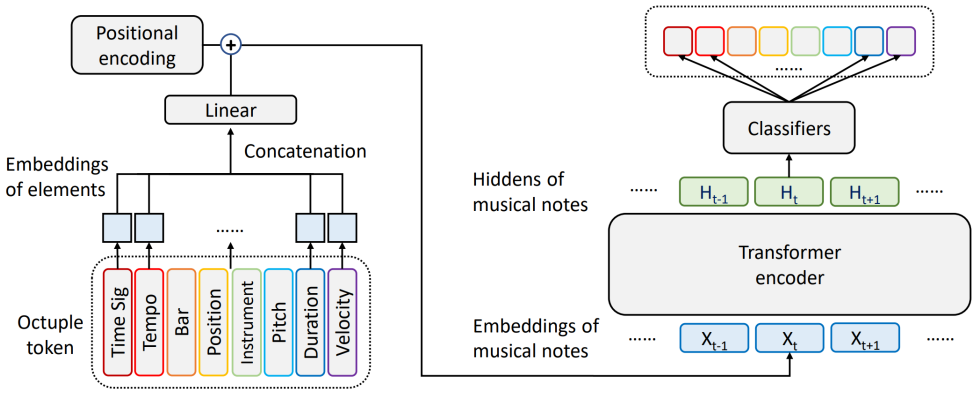

Figure 2.2 MusicBERT 구조[65]

라서 이러한 차이점을 반영하여 하나의 노트가 여러 가지 정보를 가지도록 나타낸다. MusicBERT에서는 하나의 노트를 그림 2.2에서 보이듯이 8가지 정보(octuple token)인 해당 노트의 박자표(time signature), 해당 노트의 템포(tempo), 해당 노트가 속한 바의 위치(bar), 바 내에서의 해당 노트의 위치(position), 해당 노트를 연주하는 악기(instrument), 해당 노트의 피치(pitch), 해당 노트를 연주하는 시간(duration), 해당 노트의 연주 세기(velocity)로 분리하여 나타낸다. 8가지 정보는 concatenation 된 후 선형 레이어와 포지션 인코딩을 거친 후 자연어 처리에서의 단어와 동일하게 사용된다.

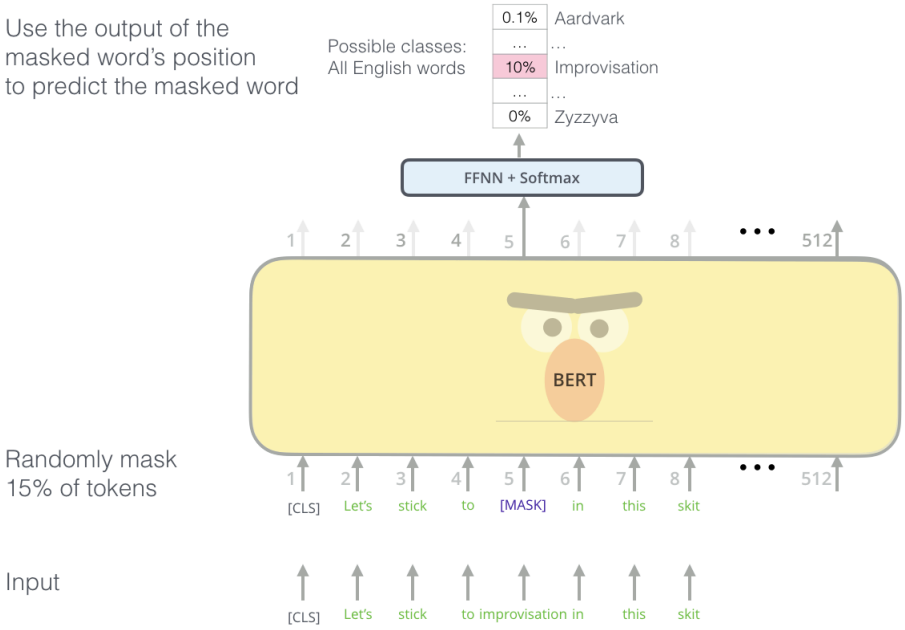

Figure 2.3 Bert Masked Language Modeling(MLM)[5]

해당 모델은 비지도 학습을 통해 곡 정보를 사전 학습(pre-training)[56]하고 그 후 사전 학습된 곡 정보를 토대로 파인 튜닝(fine-tuning)하며 다운 스트림(downstream) 과업으로 음악 장르 분류를 한다. 사전 학습 방법으로는 BERT에서 사용한 방식인 masked language modeling(MLM) 방법을 사용했다. 해당 방법은 비지도 학습 방법

(unsupervised learning)의 하나인 자기 지도 학습 방법(self-supervised learning)이다. 해당 방법을 그림 2.3에 나타냈다. 특정 단어(음표)를 마스킹하여 없앤 후 마스킹한 단어를 주위 정보들로 예측하며 예측한 단어와 실제 단어와의 에러를 통해 모델을 학습하는 방법이다. 특정 단어를 랜덤하게 15% 선택한 후 선택한 단어 중 80%는 [Mask] 토큰으로 변경하고, 10%는 그대로 두고, 나머지 10%는 랜덤한 단어로 변경한다. 랜덤한 단어로 변경하는 이유는 학습 데이터셋에서 보지 못한 단어가 테스트 데이터셋에서 나타나는 경우를 대비하기 위함이라고 한다. 그림 2.4에 MusicBERT에서 사용한 마스킹 방법을 볼 수 있다. MusicBERT에서는 노트를 BERT와 같은 방식으로 선택 후 노트에 대한

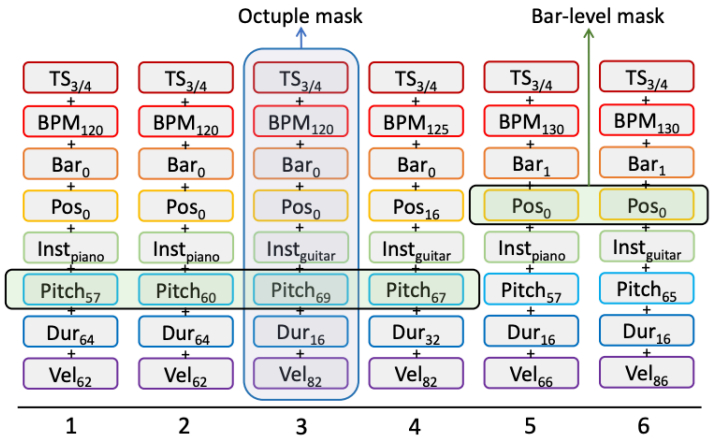

Figure 2.4 MusicBERT 마스킹 방법[5]

모든 8가지 정보를 마스킹하는 방법(octuple masking), 같은 바 안에서 동일한 토큰 형태 중 같은 값을 가지는 토큰을 같이 마스킹하는 방법(bar level masking)을 사용한다. 본 연구에서는 첫 번째 마스킹 방법을 사용하여 모델을 사전 학습했다. 사전 학습한 모델의 출력을 곡 정보로써 사용한다.

세 번째 모델은 mixture of convolutional neural network (Mixture of CNN)[64]이다. 그림 2.5에 모델 구조를 나타냈다. 해당 모델은 오디오 데이터를 멜 스펙트로그램으

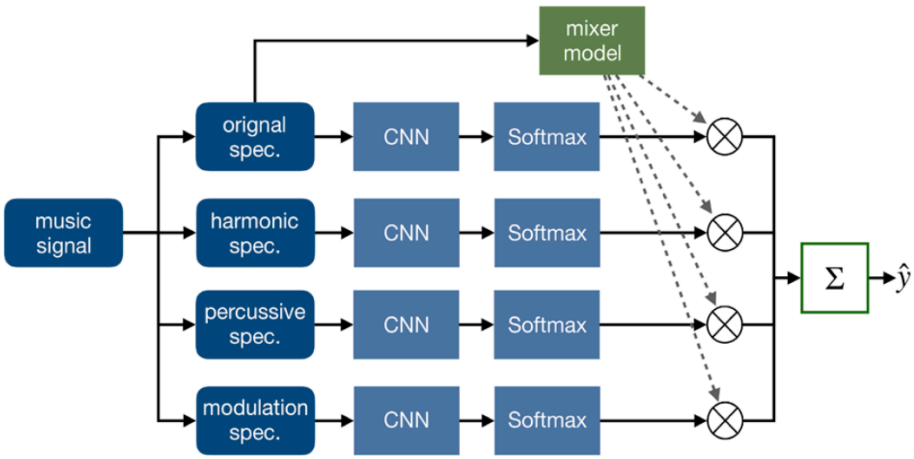

Figure 2.5 Mixture of CNN 구조[64]

로 변환하여 데이터로 활용한다. 멜 스펙트로그램으로 변환시키기 위해서 소리에 푸리에 변환(Fourier transform)을 적용하여 각 진동수 성분이 해당 소리에 얼마나 들어있는지 추출한다. 하지만 소리에 전체적으로 푸리에 변환을 적용하면 시간 정보가 사라진다. 따라서 소리를 작게(0.01초 수준) 잘라서 각 작은 조각에 푸리에 변환을 적용한 후 이 결과를 시간축으로 쌓으면 시간 정보의 삭제 없이 해당 소리를 표현할 수 있다. 이렇게 소리에 국소 푸리에 변환(short time Fourier transform, STFT)을 적용하여 나온 결과가 스펙트로그램이다. 즉, 스펙트로그램(spectrogram)은 시간에 따른 각 주파수의 음향에너지를 시각화한 것이다. 그림 2.6은 스펙트로그램의 예시이다. X축은 시간이고 Y축은 진동수, 색깔로 나타낸 Z축은 진폭을 나타낸다. 멜 스펙트로그램은 이러한 스펙트로그램에서 멜 스케일(mel-scale)을 적용하여 변환한 것이다. 사람은 저주파일 때 소리를 더 민감하게 인지하고 고주파로 갈수록 둔감하게 소리를 인지하는 특성이 있다. 이러한 특성을 반영하여 주파수에 로그 스케일을 적용한 것이 멜 스케일이다.

소리를 시각화하여 이미지 형태로 사용할 수 있으므로 딥러닝 모델에서 이미지 형태의 데이터에 알맞은 모델 구조인 콘볼루션 신경망[31]과 함께 자주 사용된다. Mixture of

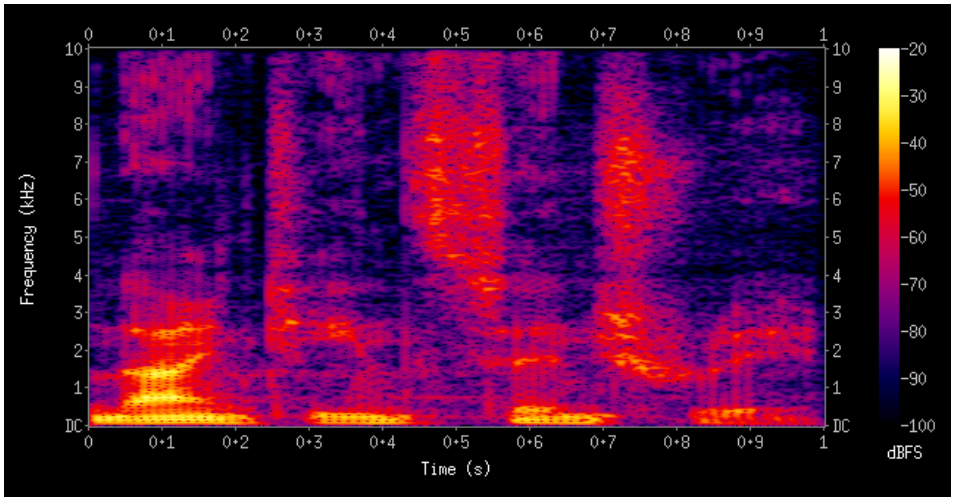

Figure 2.6 스펙트로그램[64]

CNN 모델은 오디오 데이터를 4가지 종류의 스펙트로그램(오리지널, 하모닉, 퍼커시브, 변조)으로 변환 후 멜 스케일을 적용하여 데이터로 활용한다.

오리지널 스펙트로그램은 기존 스펙트로그램(full power spectrogram)을 지칭하는 것이고 하모닉 스펙트로그램과 퍼커시브 스펙트로그램은 librosa 오픈 라이브러리[41]의 harmonic percussive source separation(HPSS) 함수를 사용하여 오리지널 스펙트로그램에서 수평적 특성과 수직적 특성을 추출한 것이다. 오디오 데이터에서 샘플링 레이트(sampling rate)는 1초당 몇 개의 데이터를 뽑아낼지를 의미한다. 윈도우 사이즈(window size)는 몇 개의 데이터를 한 구간으로 정의할지를 뜻하며 홉 사이즈(hop size)는 몇 개의 데이터를 이동해가며 윈도우 사이즈를 적용할지를 뜻한다. 이를 바탕으로 오리지널 스펙트로그램, 하모닉 스펙트로그램, 퍼커시브 스펙트로그램의 차원을 계산할 수 있다. 예를 들어, 샘플링 레이트가 1Hz이고 오디오의 길이가 10초이면 총 데이터의 수는 10개가 된다. 이 데이터를 윈도우 사이즈와 홉 사이즈를 2로 한다면 10개의 데이터는 5개의 데이터가 된다. 이때, 홉 사이즈를 1로 바꾼다면 데이터는 9개가 된다. 또한, 반대로 적용할 경우 해당 값들을 통해 얼만큼의 시간에 대한 데이터인지 알 수 있다. 기존 논문

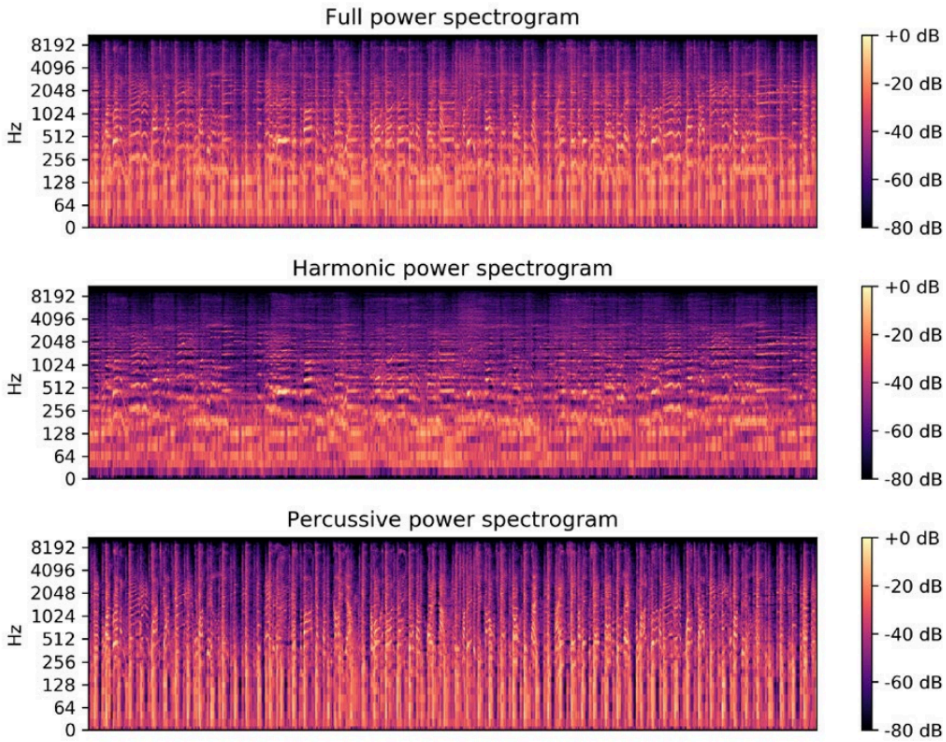

Figure 2.7 오리지널, 하모닉, 퍼커시브 스펙트로그램[64]

에서 오디오 데이터를 스펙트로그램으로 변환할 때 샘플링 레이트를 22,050Hz, 윈도우 사이즈를 1,024, 홉 사이즈를 512로 적용하고 30초 길이의 오디오 데이터를 사용했다. 따라서, 오리지널 스펙트로그램, 하모닉 스펙트로그램, 퍼커시브 스펙트로그램의 차원은 $\in \mathbb{R}^{513\times1292}$ 이다. 그림 2.7에 오리지널 스펙트로그램, 하모닉 스펙트로그램, 퍼커시브 스펙트로그램을 시각화한 예시가 있다. 변조 스펙트로그램(modulation spectrogram)은 오디오 데이터에 이산 푸리에 변환(discrete Fourier transform)을 적용한 것이다. 이렇게 변환된 4가지 스펙트로그램을 다시 멜 스펙트로그램으로 변환하여 데이터로 활용한다.

4가지 종류의 스펙트로그램을 각각 입력값으로 사용하는 4가지 모델을 지도 학습

방식으로 각각 학습한다. 이렇게 학습한 4가지 모델의 결과를 섞어 최종 장르 분류에 사용한다. 이때, 순환 신경망(recurrent neural network, RNN)[51]을 통해 4가지 모델의 결과를 섞는 가중치를 학습한다. 학습한 가중치를 바탕으로 4가지 모델의 출력을 적절히 섞은 후 최종 장르 분류에 사용한다. 제안 기법 적용 시에는 4가지 모델의 섞기 전 결과를 concatenation하여 최종 곡 정보로써 사용한다.

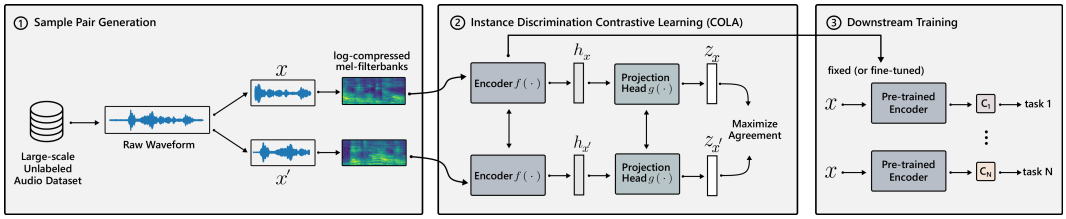

Figure 2.8 CoLA 구조[52]

마지막 네 번째 모델은 contrastive learning of general-purpose audio representations(CoLA)[52]이다. 그림 2.8에 모델의 구조를 나타냈다. CoLA에서도 앞선 Mixture of cnn과 마찬가지로 입력 데이터 형태로 멜 스펙트로그램(mel-spectrogram)을 사용한다.

CoLA는 비지도 학습 방법의 하나인 대조적 학습(contrastive learning)[12]을 통해 사전 학습한다. 대조적 학습은 데이터가 의미상으로 유사한 데이터일 경우 representation도 유사하게 학습되어야 하며 유사하지 않을 경우 representation도 유사하지 않게 학습해야 된다는 개념에서 출발한 학습 방법이다. 사람이 고양이를 본 후 호랑이를 봤을 때 고양이와 유사하다고 생각하고 상어를 봤을 때 고양이와 유사하지 않다고 생각하는 것과 같이 의미적 유사성을 고려하는 학습 방법이라고 볼 수 있다. 해당 모델에서는 같은 클립에서 나온 곡 새그먼트끼리 positive example로 정의한다. Positive example은 대조적 학습에서 유사하게 representation이 학습되길 바라는 데이터 쌍을 나타낸다. 다른 클립에서 나온 곡끼리는 negative example로 정의한다. Negative example은 rep-

resentation이 유사하지 않게 학습되길 바라는 데이터 쌍이다. 하나의 데이터를 기준으로 positive인 곡들과의 유사성과 negative인 곡들과의 유사성을 비교하여 negative인 곡들과의 유사성이 더 높을 경우 손실(loss)이 발생한다. 발생한 손실을 통해 학습한다. 유사도는 식 2.1을 통해 계산된다. $x \in \mathbb{R}^{N \times T}$로, N은 데이터를 멜 스펙트로그램으로 변환할 때 적용한 진동수 구간 수이고 T는 시간을 나타낸다. h는 입력값 x가 인코더인 f를 거친 잠재 representation이다. h의 차원은 $h \in \mathbb{R}^{d}$이고 d는 사전에 정의된 임베딩 차원이다. W는 학습되는 파라미터로 오디오 새그먼트인 x와 $x^{'}$의 유사도를 계산하는 행렬이다.

$$s(x, x^{'}) = g(f(x))^{T} W g(f(x^{'})) \tag{2.1}$$

$$s = similarity\,, x\,, x^{'} = audio\,segments$$
$$f = encoder\,, g = shallow\,neural\,network\,, W = learnable\,parameter$$

해당 모델에서는 인코더(f)로 EfficientNet-B0[54]을 사용했다. EfficientNet은 콘볼루션 신경망에서 한정된 자원으로 최대의 효율을 내기 위해 모델 스케일링(깊이, 너비(필터의 개수), 해상도)을 시스템적으로 분석하여 새로운 스케일링 방법인 혼합 스케일링(compound scaling)을 적용한 모델이다. 콘볼루션 신경망에서 모델 스케일링 변수인 깊이, 너비, 해상도는 다음과 같은 영향을 준다. 네트워크의 깊이가 증가할수록 모델의 capacity는 커지고 더 복잡한 특성을 잡아낼 수 있지만 기울기 소실 문제가 발생하여 학습이 어려워진다. 필터의 개수인 너비가 증가할수록 미세한 정보를 더 많이 담을 수 있어 정확도가 높아지지만, 계산량이 제곱에 비례하여 증가하는 문제가 있다. 해상도를 키울수록 더 세부적인 특성을 학습할 수 있지만, 너비와 마찬가지로 계산량이 제곱에 비례하여 증가한다. 공통으로 세 가지 스케일링 모두 모델의 크기가 커짐에 따라 얻는 정확도 증가량이 점점 더 적어진다. 또한, 세 가지 스케일링 변수는 서로 영향을 준다. 직관적으로 더 높은 해상도의 이미지에 대해서는 네트워크를 깊게 만들어서 더

넓은 영역에 걸쳐 있는 특성을 더 잘 잡아내는 것이 유리하다. 또한, 해상도가 높을수록 세부적인 내용을 많이 담고 있으므로 이를 다양한 필터로 분석하여 세부적인 내용을 잡아낼 수 있도록 필터의 개수인 너비를 증가시키는 것이 유리하다. 따라서 스케일링 변수를 각각 독립적으로 결정한 다른 모델들과는 다르게 EfficientNet에서는 세 가지 스케일링 변수를 동일한 비율로 움직이게 하였다. 이러한 방법으로 EfficientNet-B0는 동일한 연산량이 필요할 때 가장 높은 성능을 보인다고 알려져있다. CoLA에서는 해당 모델을 인코더로 사용한다.

네 가지 모델 모두에 대해서 기존 논문을 바탕으로 모델의 성능을 재생산(reproducing) 후 제안 기법을 적용했다.

2.2 그래프 뉴럴 네트워크

그래프 뉴럴 네트워크는 그래프 구조를 활용하는 과업에 사용되는 신경망이다. 그래프에서 노드 사이의 관계성을 모델링한다. 무작위적 관계를 모델링 할 수 있고 입력 순서와 무관하게 같은 출력값을 내보내는 그래프 뉴럴 네트워크의 inductive bias에 의해 그래프를 구성하는 요소 중 임의의 관계 구조를 모델링하는 것을 가능케 해준다[7]. Inductive bias란 학습 데이터에서 보지 못한 데이터에 대해서도 적절한 귀납적 추론이 가능하게 하도록 알고리즘(모델)이 가지고 있는 가정들의 집합을 말한다. 데이터 내 지역적 관계를 모델링 하기에 적합하고 같은 지역 정보가 들어오면 같은 결과를 내보내는 inductive bias를 가지는 콘볼루션 신경망[31]과 시계열적 관계를 모델링하기에 적합하고 같은 순서로 입력이 들어오면 같은 출력을 내보내는 inductive bias를 가지는 순환 신경망[51]과 다른 점이다. 이러한 가정들이 맞으면 모델이 좋은 성능을 보이지만 가정이 맞지 않으면 매우 약한 모습을 보여준다. 그래프에서 노드와 간선의 입력 순서는 무작위적이고 모델의 출력과 상관이 없어야 한다. 또한, 이미지와 달리 노드 내에서

지역적 정보를 모델링하는 것보다 노드끼리의 관계성을 모델링하는 것이 중요하다. 즉, 타겟 노드가 어떤 노드들과 이웃 관계를 형성하는지가 중요하고 각 타겟 노드의 입력 순서나 연결된 이웃 노드의 순서와 무관한 출력값을 가져야 한다. 따라서 기존의 모델들보다 그래프 뉴럴 네트워크가 본 연구의 제안 기법의 모델로 더 적합하다.

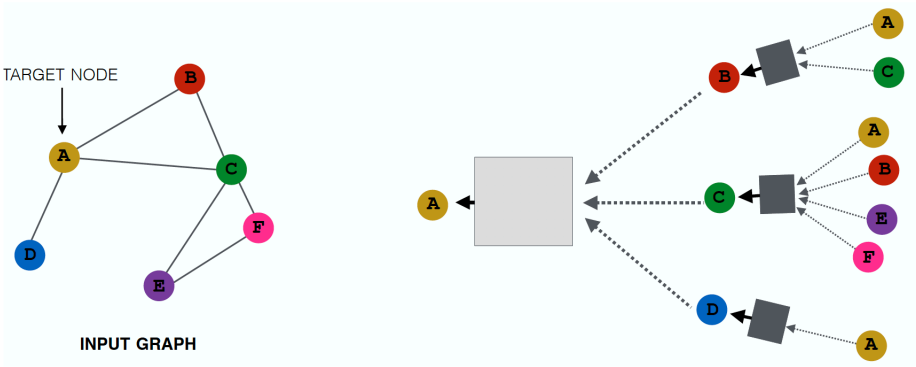

Figure 2.9 그래프 뉴럴 네트워크의 타겟 노드 업데이트 방법[1]

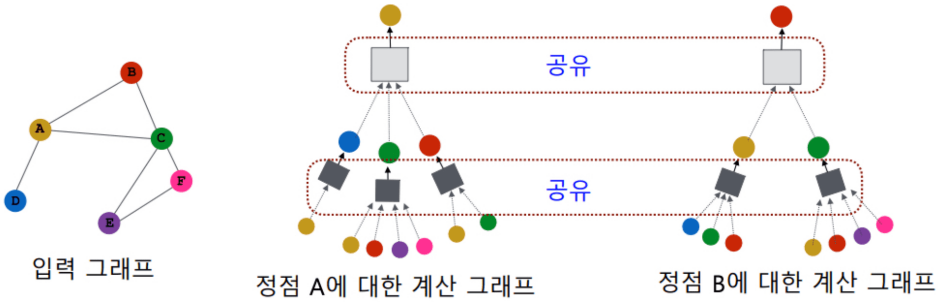

Figure 2.10 그래프 뉴럴 네트워크 계산 그래프[3]

그래프 뉴럴 네트워크는 이웃 노드들의 정보를 집계하는 것을 반복하여 타겟 노드의 임베딩을 얻는다. 반복 횟수는 모델의 레이어의 수와 같다. 그림 2.9는 레이어가 2층일 때 타겟 노드 A의 임베딩을 어떤 노드들의 정보를 집계해 업데이트하는지에 대한

그림이다. 그림에서 대상이 되는 노드 A의 임베딩을 얻기 위해 A의 이웃인 B,C,D의 정보를 집계한다. 이웃인 B,C,D의 임베딩을 얻기 위해 각 노드의 이웃에 대한 정보도 집계한다. 이때, 타겟 노드인 A도 B,C,D의 이웃 노드로서 포함된 것을 알 수 있다.

그래프에서는 노드마다 이웃들의 정보와 수가 각기 다르다. 따라서 각각의 노드마다 집계되는 정보들이 다르다. 그림 2.10은 다른 정보들이 어떻게 집계되는지를 나타낸 그림으로 각 노드에 대한 계산 그래프를 나타내고 있다. 노드별 집계 정보가 다를지라도 층별 집계 함수는 공유된다. 타겟 노드가 A인 경우와 타겟 노드가 B인 경우 모두 0층에서 1층으로 갈 때의 집계 함수가 같고 1층에서 2층으로 가는 집계 함수도 같다. 같은 집계 함수를 사용하지만 각 노드마다 연결성이 다르므로 입력의 수가 다를 수 있다. 즉, 집계 함수는 서로 다른 구조의 계산 그래프를 처리할 수 있어야 한다[3].

그래프 뉴럴 네트워크에서는 다양한 집계 함수를 사용한다. 초기에는 집계 함수로 단순히 이웃의 정보를 현재 노드가 가진 이웃의 수로 정규화한 뒤 평균하여 타겟 노드와 더하는 방법을 사용했다(식 2.2). h_v^k는 v 노드의 k층일 때의 은닉(hidden) representation을 의미한다. h_v^k의 차원은 $h \in \mathbb{R}^{d_k}$으로 d_k는 k층의 은닉 차원을 뜻한다. $N(v)$는 v 노드의 이웃 노드의 집합을 의미한다. $|\cdot|$은 $\cdot$ 집합의 수를 뜻한다. W_k는 k층에서의 이웃 노드의 정보를 집계하는 학습 가능한 파라미터를 의미하며 B_k는 k층에서의 자신의 노드의 정보를 집계하는 학습 가능한 파라미터를 의미한다. σ는 활성화 함수(activation function)를 의미하고 일반적으로 ReLU(rectified linear unit)를 사용한다.

$$h_v^k = \sigma(W_k \Sigma_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}) \tag{2.2}$$

$$h = hidden\, representation\,, v = target\, node\,, k = layer\,, W, B = learnable\, parameter$$
$$\sigma = activation\, function\,, N(v) = set\, of\, neighbor\, node\, of\, v$$

그 이후 노드 중 많은 이웃 노드를 가진 노드의 임베딩이 모델의 파라미터 학습에 많은 영향을 끼쳐 노드의 임베딩이 서로 유사해지는 획일화(over-smoothing) 문제가 발생하였다. 이러한 획일화 문제를 해결하기 위해 타겟 노드의 이웃 노드의 수에 추가로 이웃 노드에 대한 이웃 노드의 수까지 포함하여 정규화를 하는 집계 함수도 등장했다[61]. 또한, 해당 집계 함수는 자신의 노드 정보를 따로 집계하지 않고, 이웃의 노드와 같이 집계된다. 식 2.3는 해당 집계 함수를 표현한 것으로 $\Sigma_{u \in N(v) \sqcup v}$ 에 의해 자기 자신의 노드도 이웃 노드와 같은 파라미터를 통해 집계되는 것을 알 수 있다. 또한, $\sqrt{|N(u)||N(v)|}$ 를 통해 타겟 노드인 v 노드에 더해서 이웃 노드의 이웃 노드 수까지 합쳐 정규화를 진행하는 것을 알 수 있다.

$$h_v^k = \sigma(W_k \Sigma_{u \in N(v) \sqcup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}}) \tag{2.3}$$

$$h = hidden\, representation\,, v = target\, node\,, u = neighbor\, node\,, k = layer$$
$$W = learnable\, parameter\,, \sigma = activation\, function\,, N(\cdot) = set\, of\, neighbor\, node\, of\, \cdot$$

[28]에서는 집계 함수로 식 2.2와 유사하나 타겟 노드의 정보를 단순히 이웃 노드의 정보들과 더하지 않고 concatenation한다. 이는 단순히 이웃 노드의 정보와 함께 더해주는 것과는 달리 타겟 노드의 정보를 이웃 노드와는 별도로 관리하고 활용한다고 볼 수 있다. 따라서, 타겟 노드 자신의 곡 정보가 중요한 정보라고 여겨지는 본 연구에서는 이웃 노드 정보를 집계하는 파라미터와 별개의 파라미터를 통해 타겟 노드의 정보를 집계하고 타겟 노드의 정보를 집계된 이웃 노드의 정보에 concatenation하여 사용하는 해당 방법을 집계 함수로써 사용한다.

2.3 관련 연구

딥러닝을 적용하기 전 장르 분류 연구에서는 곡에서 특정 특성을 추출하고 추출한 특성을 바탕으로 장르를 분류하는 연구가 주를 이뤘다. [57]에서는 오디오 데이터에서 음색, 리듬, 피치의 특성을 추출하여 추출한 특성들끼리의 유사도를 계산 후 유사도를 토대로 장르를 분류하였다. [34]에서는 오디오를 히스토그램으로 표현한 뒤 표현한 히스토그램끼리 유사도를 데이터로 활용해 머신러닝 방법인 서포트 벡터 머신(support vector machine, SVM)[15]과 선형 판별 분석(linear discriminant analysis, LDA)[55]을 사용하여 장르를 분류한다. 심볼릭 데이터를 활용한 연구[11]에서는 미디 데이터 간의 정규화된 압축 거리(normalized compression distance, NCD)[32]를 바탕으로 K-최근접 이웃 알고리즘(k-nearest neighbor, KNN)[47]을 사용해 장르를 분류한다. 또한, 동일한 연구에서 미디 데이터를 오디오 데이터로 변환하여 미디 데이터와 변환된 오디오를 모두 사용하여 장르를 분류한다. 이처럼 딥러닝 적용 전 연구는 장르를 분류할 때 적절하다고 판단되는 특성을 추출하고 추출한 특성을 기반으로 모델이 장르를 분류했다.

초기 딥러닝 연구에서는 오디오 데이터를 벡터화시킬 수 있는 mel-frequency cepstral coefficient(MFCC)[37]나 스펙트로그램을 활용해 장르를 분류하였다[35, 22]. 딥러닝에서 데이터를 바로 벡터화시킬 수 있다는 것은 모델을 학습시킬 수 있다는 의미와 같으므로 많은 음악 관련 연구가 오디오를 활용했다. 이러한 연구들은 대부분 지도 학습을 통해 모델을 학습했으며 추후에 라벨링 없이도 학습이 가능한 비지도 학습이 많은 관심을 끌게 됐다. [13]은 오디오의 특정 부분을 마스킹하여 마스킹한 부분을 예측하여 모델을 학습하는 비지도 학습 중 가장 많이 사용되는 자기 지도 학습 방법(self-supervised learning)을 통해 모델을 학습한다. 하지만 이런 자기 지도 학습은 데이터끼리의 의미론적 유사성을 학습하지 못하는 문제가 존재한다. 이러한 문제점을 해결하기 위해 대조적 학습이 등장했다. 음악에서는 의미론적 유사성이 중요하다고 여겨지므로

최근에는 이러한 대조적 학습을 통해 데이터끼리의 의미론적 유사성을 바탕으로 음악 representation을 학습하는 연구가 많이 진행됐다[52, 24, 59, 40].

곡을 심볼릭 형태로 표현한 후 음악 representation을 학습하는 연구도 존재한다. 오디오를 대표적인 심볼릭 음악 representation인 피아노롤 형태로 변환하여 작곡가 분류에 활용하는 연구[58]가 있다. 피아노롤 형태가 아닌 MIDI 형태의 심볼릭 데이터를 활용하는 연구도 존재한다. MIDI 데이터를 활용하는 대다수의 연구는 자연어 처리 (natural language processing) 분야에서 사용됐던 방법을 많이 차용하여 음악 도메인에 적용한다. 자연어 처리 분야에서 단어의 representation을 학습하기 위해 사용한 skip-gram[43]을 통해 멜로디 representation을 학습한 후 학습한 멜로디 정보를 바탕으로 장르 분류에 활용하는 연구[29]도 있다. 또한, 여러 연구가 자연어 처리에서의 단위인 단어를 음악에서의 단위인 노트에 매칭시켜 음악 representation 학습을 시도한다. [14]에서는 노트를 4가지 정보인 해당 노트의 피치, 해당 노트가 특정 바에서의 시작 노트 여부, 해당 노트의 길이, 바 내에서의 해당 노트의 위치로 변환한 후 BERT[17]에서 사용한 자기 지도 학습 방법의 하나인 masked language modeling(MLM)을 사용하여 음악 representation을 학습한다. [36]에서는 곡을 멜로디와 하모니 두 부분으로 나누어 representation을 학습하며 해당 연구 역시도 MLM 방법을 통해 representation을 학습한다. 이렇게 학습한 멜로디 정보와 하모니 정보를 합쳐 최종 곡 정보로 활용한다. 최근에는 오디오 데이터나 심볼릭 데이터에 추가로 텍스트 데이터와 같은 다른 형태의 데이터를 추가 활용하는 연구도 존재한다[39].

이처럼 딥러닝 적용 후 대부분의 연구는 음악적 특성인 곡의 정보를 학습하여 장르 분류에 활용하고 있다. 이러한 음악 정보를 학습한 모델들에 제안 기법인 그래프 뉴럴 네트워크를 활용해 아티스트 정보를 추가 활용할 수 있다. 해당 제안 기법은 특정 데이터 형태에 대해 제한 없이 음악 representation을 생성할 수 있는 모든 모델에 적용

가능하다.

곡만을 활용한 연구 이외 추가로 아티스트 정보를 활용한 연구 또한 소수 존재한다. [45] 에서는 같은 아티스트일 경우 positive example, 다른 아티스트일 경우 negative example로 정의한 후 대조적 학습을 통해 음악 representation을 학습한다. [6] 에서도 유사한 방식으로 모델을 학습하나 아티스트 정보뿐 아니라 릴리즈 , 레코드 레이블을 활용하여 positive example, negative example을 정의한 후 대조적 학습을 통해 모델을 학습한다. 두 방법 모두 아티스트 정보를 대조적 학습에서 데이터끼리의 관계를 정할 때 사용한다. 본 연구에서는 아티스트 정보를 그래프에서 이웃 노드를 정의할 때 사용한다.

제 3 장 아티스트 정보를 효과적으로 활용할 수 있는 그래프 뉴럴 네트워크 기반 플러그 앤 플레이 프레임워크

3.1 제안 기법 구조

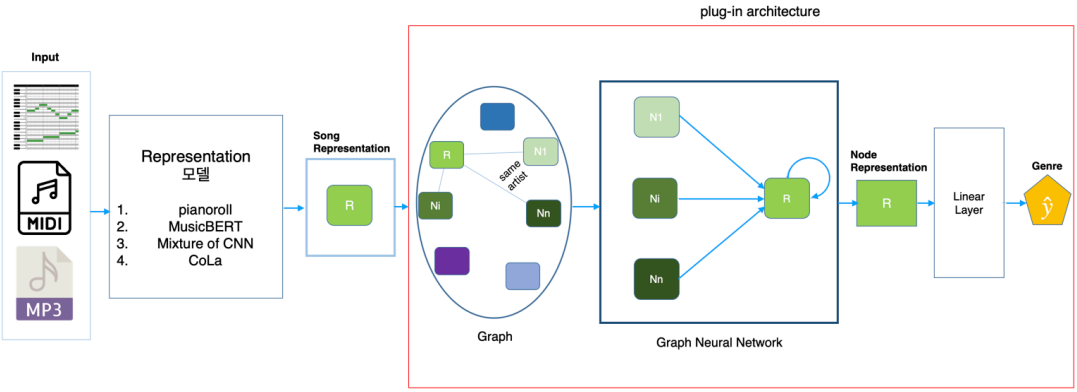

Figure 3.1 제안 기법 구조

그림 3.1은 제안 기법의 구조를 설명한 그림이다. 각 representation 모델에 맞는 음악 데이터 형태를 입력값으로 모델에 제공한다. 각 모델은 모델이 학습한 것을 토대로 입력되는 데이터를 해석하고 가공하여 표현한 결과인 곡 정보(song representation)를 출력한다(2.1절 참고). 출력한 곡 정보는 노드로 정의하였다. 그래프 뉴럴 네트워크는 이웃 노드의 정보를 집계하며 모델을 학습하므로 어떤 방식으로 이웃 노드들이 정의되는지가 매우 중요하다. 아티스트 정보만을 활용했을 때 네 가지 데이터셋에서 높은 성능을 보인 점을 근거로 하여 두 노드가 같은 아티스트일 경우 두 노드를 이웃 노드로 정의하였다. 형성된 이웃 곡 정보들은 그래프 뉴럴 네트워크를 통해 타겟 노드의 정보와 함께 다음 층의 타켓 노드의 정보로 업데이트된다. 각각의 타겟 노드에 대해 집계된 정보는 노드 정보(node representation)로 정의되고 노드 정보는 분류기인 선형

레이어를 통과해 최종 장르를 분류한다.

제안 기법을 적용하기 위해 사전 학습이 필요한 모델들인 MusicBERT와 CoLA는 먼저 사전 학습을 진행한 후 제안 기법을 적용했다. MusicBERT는 심볼릭 데이터셋인 Lakh MIDI 데이터셋을 사용하여 사전 학습하였다. MusicBERT는 BERT와 마찬가지로 최대 1,024개의 노트까지 입력값으로 사용할 수 있으므로 1,024개 보다 많은 노트를 가지는 곡들에 대해서는 3 ∼ 4개로 랜덤하게 샘플링하여 사용하였다. 사전 학습 후 각 곡에 대한 모델의 출력값을 제안 기법 적용 시 곡 정보로써 노드로 활용했다. CoLA는 MTG-Jamendo 데이터셋을 사용했다. 해당 데이터셋은 태깅 데이터셋이므로 장르 태그가 없는 곡들은 사용하지 않았다. 총 87개의 장르 중 빈도수 기준 상위 15개의 장르를 포함하는 데이터를 장르 분류 데이터셋으로 사용했다. 15개의 장르를 포함하지 않는 나머지 데이터는 대조적 학습 데이터로 활용했다. 대조적 학습을 통해 사전 학습 후 각 곡에 대한 모델의 출력값을 제안 기법 적용시 곡 정보로써 노드로 활용했다. 피아노롤 모델은 지도 학습 방법으로 학습하였으므로 사전 학습을 진행하지 않았고 Lakh pianoroll 데이터셋 자체를 곡 정보로써 노드로 활용했다. Mixture of CNN 또한 지도 학습 방법으로 학습하였으므로 사전 학습을 진행하지 않았다. 또한, 모델 특성상 4개 모델의 출력값을 RNN 모델을 통해 섞어서 최종적으로 장르를 분류하므로 각각의 모델을 학습한 후 섞기 전 4가지 모델의 출력값을 concatenation하여 곡 정보로써 노드로 활용했다.

3.2 베이스라인 모델

베이스라인 모델은 크게 3가지로 나뉜다. 제안 기법이 곡의 정보와 아티스트 정보를 함께 활용하므로 곡의 정보만을 사용한 결과, 아티스트 정보만을 사용한 결과를 첫 번째, 두 번째 베이스라인으로 정했다. 곡의 정보만을 사용한 실험은 기존 representation 모델들의 결과들이다. 제안 기법과의 비교 시 해당 논문에 게재된 실험 결과의 성능을 사용했다. 아티스트 정보만을 사용한 실험은 의사결정나무(decision tree) 모델[48]의 결과를 사용했다. 의사결정 나무는 주어진 입력값들의 조합에 대한 의사결정규칙(rule)에 따라 입력값을 분류하는 모형이다. 본 연구에서는 학습 데이터 내의 각 아티스트의 보유 장르 중 가장 빈도가 높은 장르로 테스트 데이터에 존재하는 동일한 아티스트의 장르를 예측하는 의사결정규칙을 토대로 예측한다. 학습 데이터셋에 존재하지 않는 아티스트가 테스트 데이터셋에 나타날 때 해당 데이터는 어떤 장르도 예측하지 않게 하여 본 연구의 동기였던 일반 사람들이 일반적으로 예측하는 방법과 동일하게 분류하고자 하였다. 곡의 정보만을 사용한 결과와 아티스트 정보만을 사용한 결과를 비교하여 기존 모델들이 일상에서 아티스트만을 보고 장르를 예측하는 것에 비해 얼마나 장르 관점에서 곡을 이해하고 있는지 알 수 있다. 본 연구에서 사용한 동일 데이터셋에 대해 진행한 선행 연구가 존재할 경우 해당 모델들을 마지막 베이스라인 모델로 정했다.

3.3 그래프 뉴럴 네트워크 모델

본 연구에서는 그래프 뉴럴 네트워크를 노드 수준 분류 과업(node-level classification)을 위해 사용한다. 모델 구조로는 기본 모델 중 하나인 GraphSAGE[28] 모델을 사용했다. 기본적인 그래프 뉴럴 네트워크 모델을 사용함으로 모델 자체의 추가된 강점을 최소화하여 순수하게 그래프 구조를 통한 아티스트 정보를 활용했을 때의 성능 향상을 관찰하기 위함이다. 기본 모델로 알려진 그래프 콘볼루션 모델(GCN)과 GraphSAGE 중 GraphSAGE를 사용한 이유는 다음과 같다. 그래프 콘볼루션 모델은 타겟 노드에 관한 정보를 이웃 노드의 정보와 함께 집계한 후 타겟 노드를 업데이트한다. 반면, GraphSAGE는 타겟 노드의 정보를 이웃 정보를 집계하는 파라미터와 별도로 분리하여 다른 파라미터를 통해 집계하고 집계된 이웃 노드 정보에 타겟 노드의 정보를 concatenation 하여 다음 층의 타겟 노드 정보로 업데이트한다. 이는 GraphSAGE 모델이 타겟 노드의 관점에서 자기 자신의 정보에 대해 GCN보다 더 많은 비중을 주며 타겟 노드를 업데이트한다고 볼 수 있다. 따라서, 타겟 노드 자기 자신의 곡 정보가 중요하다고 생각되는 음악 장르 분류에서 타겟 노드의 정보에 대한 집계 함수를 이웃 노드와 분리하여 별도로 학습하고 집계된 이웃 정보에 concatenation하여 타겟 노드의 곡 정보를 이웃 노드의 정보와 분리하여 활용하는 GraphSAGE 모델을 본 연구의 그래프 뉴럴 네트워크로 선택하였다. 식 3.1에 해당 모델이 이웃 노드를 집계하는 방법을 나타냈다. 대부분의 변수가 식 2.2와 동일하다.

$$
\begin{aligned}
h_v^k &= \sigma([W_k AGG(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_v^{k-1}]) \\
AGG &= \Sigma_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}
\end{aligned}
\tag{3.1}
$$

$$h = hidden\,representation\,, v\,, u = node\,, k = layer\,, W, B = learnable\,parameter$$

$AGG = aggregation\,function\,, \sigma = activation\,function\,, N(v) = set\,of\,neighbor\,node\,of\,v$

그래프 뉴럴 네트워크에서는 모델 학습 시 이웃 노드를 최대한 활용하여 파라미터들을 학습하기도 하고 사용 이웃 노드의 수를 정한 후 학습마다 정의된 수만큼의 이웃 노드의 정보만을 사용해 파라미터를 학습하기도 한다. 본 연구에서는 모든 이웃의 노드 수를 사용하려고 하였지만, MusicBERT와 CoLA의 경우 곡 정보의 차원이 매우 커 모든 이웃 노드를 사용할 경우 메모리 용량을 넘어 대부분의 이웃 노드를 사용할 수 없었다. 따라서 학습 시에는 이웃 노드를 샘플링하는 neighbor sampler를 사용해 집계할 이웃 노드의 수를 매 학습 에포크마다 특정 개수로 제한하여 사용했다. 모든 모델에 대해 학습을 진행할 때 8 $\sim$ 12개 사이의 이웃 노드의 수를 정하여 학습마다 정해진 이웃 노드의 수만큼 랜덤으로 샘플링 해 사용했다. 추론 시에는 배치 사이즈를 1로 줄인 후 최대한 많은 이웃 노드 수를 사용했다. 한편, 여러 가지 장르를 가지는 아티스트의 경우 대다수를 차지하는 장르에 의해 그래프 뉴럴 네트워크에서 발생하는 문제인 획일화 문제(over-smoothing)가 생길 수 있다. 이를 완화하기 위해 DropEdge[50] 기법을 적용한 실험도 진행했다. DropEdge는 모델 학습 도중 학습마다 연결된 간선을 랜덤으로 30%를 제거하는 기법이다. 이렇게 학습마다 연결된 이웃 노드가 랜덤하게 달라지므로 DropEdge는 데이터 증강기법과 동일한 효과인 모델의 강건성을 높인다고 알려져있다. DropEdge를 적용할 때와 적용하지 않았을 때에 대해 비교 실험을 진행했다.

3.4 아티스트 정보 활용 방법

본 연구에서는 아티스트 정보를 직접적으로 사용한 방법과 그래프 구조를 사용하여 아티스트 정보를 노드 간의 관계를 정의하는 간접적 방법으로 사용한 방법을 비교한다. 아티스트 정보를 직접적으로 사용한 방법은 모델 학습 시 각 아티스트마다 임베딩을 학습하였다. 학습한 임베딩은 곡 정보에 concatenation되어 분류기에 입력된다. 아티스트 정보가 직접적으로 곡의 정보에 합쳐져 분류기에 입력되므로 아티스트 정보를 직접적으로 사용한다고 볼 수 있다. 아티스트 정보를 간접적으로 사용한 방법은 3.1절에서 언급한 방법으로 그래프의 노드들을 정의한다. 그 후 같은 아티스트인 노드끼리 간선으로 연결한다. 즉, 아티스트 정보를 이웃 노드를 정의하는 데 사용한다. 해당 방법은 아티스트의 정보를 곡의 정보에 직접적으로 추가해 사용하지도 않고 아티스트 정보를 따로 학습하지도 않는다. 단지 그래프 구조를 만드는 데 활용한다. 따라서 해당 방법을 아티스트 정보를 간접적으로 사용한다고 정의하였다.

제 4 장 실험 결과

4.1 데이터셋

본 연구에서는 총 네 가지 오픈 데이터셋을 사용했다. 첫 번째 데이터셋은 Free Music

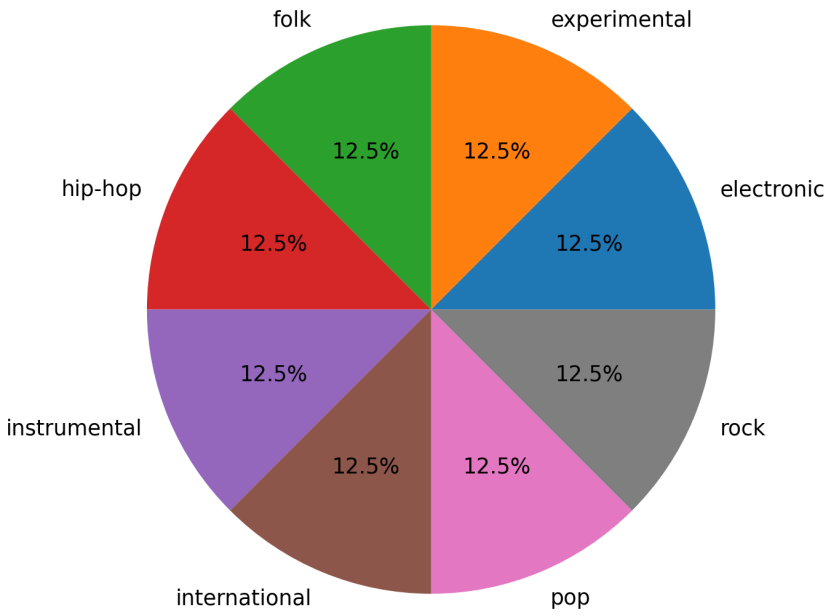

Figure 4.1 FMA Small 데이터셋 장르 구성

Archive(FMA)[16]이다. 해당 데이터셋은 Full, Large, Medium, Small 총 4가지가 존재한다. 본 연구에서는 음악 장르 분류 과업에 자주 사용되는 Small 데이터셋을 사용했다. 해당 데이터셋은 총 8,000개의 오디오 데이터가 존재하고 장르는 electronic, pop, rock, experimental, folk, instrumental, international, hip-hop으로 총 8가지 장르가

존재한다(그림 4.1). 총 아티스트의 수는 2,307명으로 아티스트 당 평균 3.5개의 곡이 존재한다. 각 데이터는 단일 라벨(single label)로 구성되어있다. 해당 데이터셋을 통해 Mixture of cnn 모델을 학습했다. 데이터셋이 학습, 검증, 테스트로 나눠 제공되므로 미리 정의된 학습, 검증, 테스트 데이터셋을 사용했다.

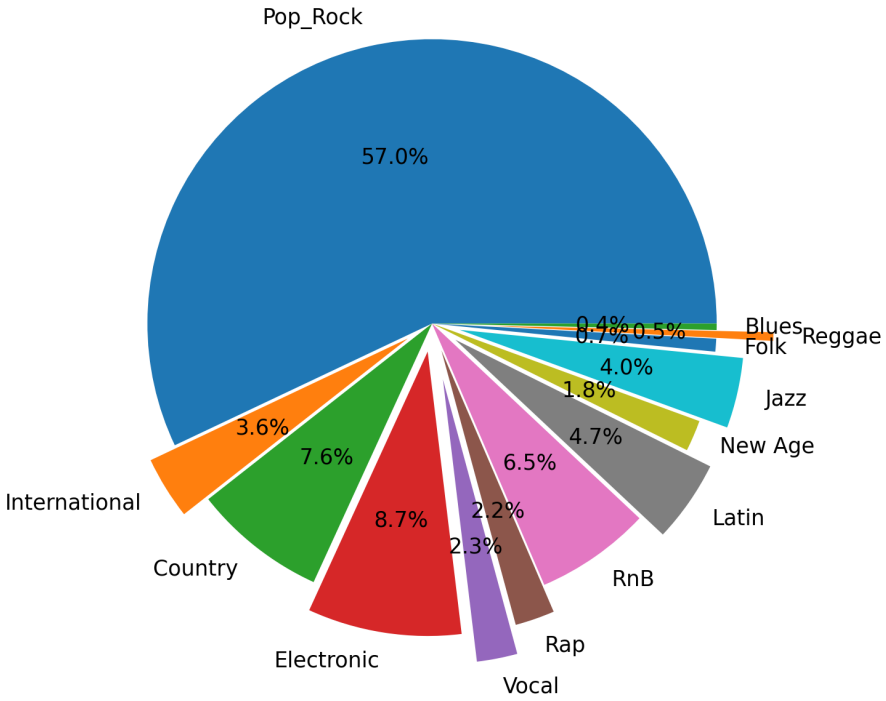

Figure 4.2 Lakh MIDI 데이터셋 장르 구성

두 번째 데이터셋은 Lakh MIDI[49] 데이터셋이다. 총 176,581개의 곡이 존재한다. 모든 데이터를 이용하여 MusicBERT를 사전 학습했다. 그 후 장르가 존재하는 22,535개의 서브 데이터셋을 사용하여 모델을 파인 튜닝했다. MusicBERT의 기존 논문과 같은 방식으로 곡당 3 ∼ 4개씩 샘플링하여 총 89,160개의 데이터를 사용했다. 총 아티스트의 수는 4,352명으로 아티스트 당 평균 5.2개의 곡이 존재한다. 한 곡당 3 ∼ 4개의 샘플링을 했으므로 동일한 아티스트의 데이터 수는 평균 20.5개가 존재한다. 해당 데이터셋 내

장르는 pop/rock, country, electronic, R&B, latin, jazz, new age, rap, international, reggae, folk, vocal, blues로 총 13가지가 존재한다(그림 4.2). 각 데이터는 다중 라벨(multi-label)로 구성되어 있다. 데이터셋을 8:1:1 비율로 나눠 학습, 검증, 테스트 데이터로 사용했다.

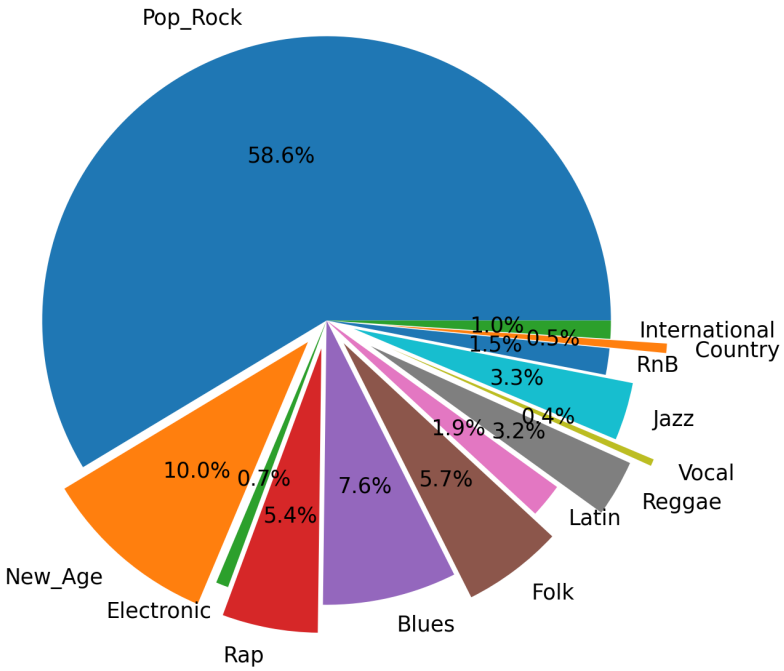

Figure 4.3 Lakh Pianoroll 데이터셋 장르 구성

세 번째 데이터셋은 Lakh MIDI 데이터셋을 피아노롤 형태로 바꾼 Lakh pianoroll[21] 데이터셋이다. 해당 데이터셋을 통해 피아노롤 모델을 학습했다. 변환 과정에서 오류가 발생하는 데이터를 제외하고 총 11,769개의 곡이 존재한다. 곡당 7 ∼ 8 개씩 샘플링하여 총 84,512개의 데이터를 사용했다. 총 아티스트의 수는 2,705명으로 아티스트 당 4.3개의 곡이 존재한다. 한 곡당 7 ∼ 8개의 샘플링을 했으므로 동일한 아티스트의 데이터 수는 평균 31.2개가 존재한다. 장르 종류는 Lakh MIDI 데이터셋과 동일하고(그림 4.3)

해당 데이터셋의 데이터 또한 다중 라벨로 구성되어 있다. 데이터셋을 8:1:1 비율로 나눠 학습, 검증, 테스트 데이터로 사용했다.

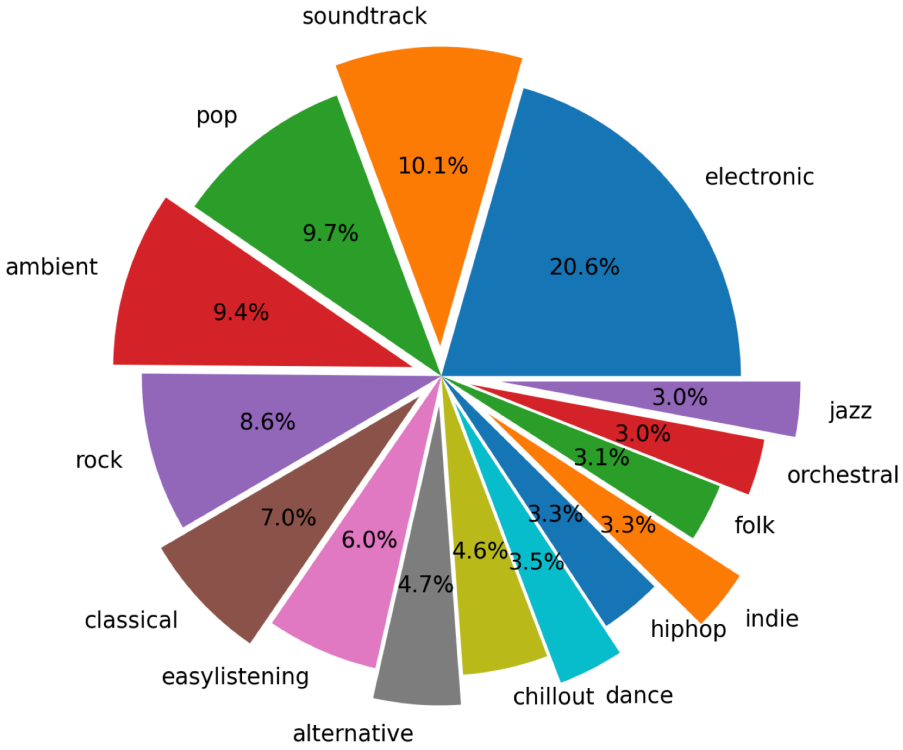

Figure 4.4 MTG-Jamendo 데이터셋 장르 구성

마지막 데이터셋은 MTG-Jamendo[9] 데이터셋이다. 해당 데이터셋은 장르, 악기, 무드/테마가 태깅된 데이터셋이므로 장르를 가지지 않는 곡들을 제거한 후 사용했다. 총 87개의 장르 중 빈도수 기준으로 상위 15개의 장르를 사용했다. 15개의 장르는 electronic, soundtrack, pop, ambient, rock, classical, easylistening, alternative, chillout, dance, hip-hop, indie, folk, orchestral, jazz이다(그림 4.4). 해당 장르를 포함하지 않는 나머지 데이터는 CoLA 모델을 사전 학습하기 위해 사용했다. 총 48,590 데이터를 장르 분류에 사용했다. 총 아티스트의 수는 3,546명으로 아티스트당 13.7개의 데이터가 존재한다. 각 데이터는 다중 라벨로 구성되어 있다. 데이터셋을 8:1:1 비율로 나눠 학습, 검증, 테스트

데이터로 사용했다.

4.2 실험 세팅

본 연구에서는 곡만의 정보만을 사용했을 때, 아티스트 정보만을 사용했을 때, 곡의 정보에 직접적으로 아티스트 정보를 추가 활용했을 때, 곡의 정보에 간접적으로 아티스트 정보를 활용했을 때에 따른 음악 장르 분류 성능을 비교한다. 성능 비교를 통해 곡의 정보와 아티스트 정보를 한 가지씩 사용했을 때에 비해 두 가지 정보를 모두 활용할 경우의 효과를 알 수 있다. 또한, 곡만의 정보를 사용한 경우와 아티스트 정보만을 사용한 경우를 비교하여 아티스트 정보를 사용하는 것이 곡의 정보에 비해 음악 장르 분류 관점에서 얼마큼의 영향력을 가지는지도 알 수 있다. 아티스트 정보를 직접적, 간접적으로 활용했을 때를 비교하여 본 연구의 제안 기법인 그래프 뉴럴 네트워크를 활용한 간접적 아티스트 정보 제공의 효과를 확인할 수 있다. 또한, 많은 그래프 뉴럴 네트워크 관련 과업에서 좋은 효과를 보인다는 DropEdge[50]를 적용한 방법과 적용하지 않은 방법과의 성능 비교를 통해 음악 데이터셋 내에서의 해당 기법의 성능을 확인할 수 있다.

다음으로 모델별 실험 세팅에 대하여 설명한다. 첫 번째 모델은 MusicBERT이다. MusicBERT는 깃헙(github)에 코드가 공개됐지만, 해당 코드는 페이스북의 fairseq[44] 오픈 라이브러리를 사용해 코드가 구성되어있다. 해당 라이브러리를 사용할 경우 그래프 뉴럴 네트워크를 적용하기 힘들다는 점 때문에 pytorch[46] 오픈 라이브러리 형식으로 코드를 변환 후 사용했다. 또한, 그래프 뉴럴 네트워크를 적용하기 위해서 pytorch geometric[25] 라이브러리를 사용했다. 해당 모델은 사전 학습 후 다운 스트림(downstream) 과업으로 장르 분류를 한다. 사전 학습 시 모델 하이퍼파라미터는 기존의 논문과 동일하게 설정하였다. 그래프 뉴럴 네트워크를 추가한 모델의 하이퍼파라미터는 표 4.1과 같다. 학습할 때는 neigbor sampler를 사용해 노드 당 5개의 이웃 노드를 사용했다. 추론 시에는 각 노드의 모든 이웃 노드를 사용하려 했지만, 메모리 부족으로

사용할 수 없어 최대 100개의 이웃 노드를 사용했다. 학습률 스케줄러로는 람다 스케줄러를 사용했고 매 에포크마다 0.96승 만큼 학습률을 감소시켰다. 학습 조기 종료(early stopping) 기법도 사용하여 검증 평가 지표가 8번의 에포크동안 증가하지 않으면 학습을 중지했다.

Table 4.1 MusicBERT 하이퍼파라미터

하이퍼파라미터	값
Num Layer	1
GNN Out Dim	3584
Dropout Rate	0.5
Number of Neighbor	5
Learning Rate	0.00003
Optimizer	Adam
Batch Size	8

평가 지표로는 실제로 참인 값을 모델이 참으로 예측한 비율인 재현율(recall)과 모델이 참으로 예측한 값들이 실제 참인 비율인 정밀도(precision)의 조화평균 값인 F1 Score를 사용한다. 또한, 본 연구에서 사용하는 대부분의 데이터셋들이 라벨 별 데이터의 수가 불균형하므로 이러한 불균형 데이터에 보다 적합하다고 알려진 micro F1 Score를 평가 지표로 사용했다. 하나의 곡에 대해 정답 장르들이 여러 개로 이뤄져 있으며, 여러 개로 구성된 장르가 각각 독립적이므로 각각의 장르에 대한 확률값을 계산하여야 한다. 따라서 손실 함수로는 다중 라벨 이진 크로스 엔트로피(multi label binary cross entropy)를 사용했다. 다중 라벨 크로스 엔트로피는 각 라벨별로 이진 교차 크로스 엔트로피(binary cross entropy, BCE)를 구한 후 평균을 취한 값이다. 식 4.1에서 $Multi\,Label\,Binary\,Cross\,Entroy\,Loss$은 다중 라벨 이진 크로스 엔트로피를 계산하는 식을 나타내며 $BCELoss(\hat{y}_i, y_i)$는 정답 라벨을 의미하는 y_i과 해당 라벨을 예측한 $\hat{y}_i$에 대해 이진 교차 크로스 엔트로피를 계산하는 것이다. 차원은 $y_i, \hat{y}_i \in \mathbb{R}^{B \times l}$

으로 B는 배치 사이즈이고 데이터셋이 다중 라벨로 구성되어있으므로 l은 라벨 수를 의미한다. N은 총 데이터 수를 의미하고 배치 단위로 학습할 경우 배치 사이즈와 동일한 의미이다.

$$
\begin{gathered}
Multi\,Label\,Binary\,Cross\,Entroy\,Loss = \Sigma_{i=1}^{n}\Sigma_{j=1}^{l} BCELoss(\hat{y_{ij}}, y_{ij}) \\
BCELoss(\hat{y}_i, y_i) = -(y_i \times \log \hat{y}_i + (1 - y_i) \times \log(1 - \hat{y}_i))
\end{gathered} \tag{4.1}
$$

$$
\begin{gathered}
i = index\,of\,data\,, y = label\,, \hat{y}_i = predicted\ label \\
l = number\,of\,labels\,, j = label\,index
\end{gathered}
$$

다음 모델은 피아노롤 모델이다. 피아노롤 모델의 기존 논문에는 음악 장르 분류 결과가 존재하지 않아 해당 모델을 베이스라인을 활용한 다른 논문의 결과를 참고했다[36]. 그래프 뉴럴 네트워크를 추가한 모델의 하이퍼파라미터는 표 4.2와 같다. 학습할 때는 노드 당 16개의 이웃 노드를 사용했고 추론 시 모든 이웃 노드의 수를 사용했다. MusicBERT와 동일한 람다 학습 스케줄러를 사용했고 매 에포크마다 0.98승 만큼 학습률을 감소시켰다. 15번의 에포크 동안 검증 평가 지표가 증가하지 않으면 학습을 조기 종료했다. 평가 지표와 손실 함수는 위의 MusicBERT와 동일한 것을 사용했다.

Table 4.2 피아노롤 모델 하이퍼파라미터

하이퍼파라미터	값
Num Layers	1
GNN Out Dims	384
Dropout Rate	0.5
Number of Neighbors	16
Learning Rate	0.00045
Optimizer	Adam
Batch Size	16

다음 모델인 Mixture of CNN은 깃헙에 코드가 공개돼 있지만 tensorflow[4] 라이

브러리로 구현돼 있어 해당 공개 코드를 기반으로 pytorch 라이브러리 형태로 변경 후 해당 모델을 사용했다. 그래프 뉴럴 네트워크를 추가한 모델의 하이퍼파라미터는 표 4.3과 같다. 학습 시 노드 당 8개의 이웃 노드를 사용했고 추론 시에는 모든 이웃 노드를 사용했다. 학습 스케줄러는 피아노롤 모델과 동일한 람다 스케줄러를 사용했다. 학습 조기 종료 제한은 10번의 에포크로 했다. 해당 모델이 학습한 데이터셋이 다른 모델들의 데이터셋과는 다르게 단일 라벨로 구성되어 있으므로 평가 지표로는 정확도(accuracy)를 사용했고 손실 함수로는 이진 크로스 엔트로피(binary cross entropy)를 사용했다.

Table 4.3 Mixture of CNN 하이퍼파라미터

하이퍼파라미터	값
Num Layers	1
GNN Out Dims	256
Dropout Rate	0.5
Number of Neighbors	8
Learning Rate	0.005
Optimizer	Adam
Batch Size	16

다음 모델은 CoLA이다. CoLA 또한 앞선 Mixture of CNN과 마찬가지로 tensorlfow 라이브러리인 것을 pytorch 라이브러리로 변환 후 사용했다. CoLA는 대조적 학습을 통해 사전 학습을 진행 후 다운 스트림(downstream) 과업으로 장르 분류를 수행하는 모델이다. 본 연구에서는 MGT-Jamendo 데이터셋 중 장르 분류에 사용하지 않는 곡들을 사용해 대조적 학습을 수행했다. 대조적 학습을 진행할 때 하이퍼파라미터는 기존 연구와 동일한 값을 사용했다. 그래프 뉴럴 네트워크를 추가한 모델의 하이퍼파라미터는 표 4.4와 같다. 학습과 추론 시 이웃 노드 사용 방식은 Mixture of CNN과 같다. 학습 스케줄러는 람다 스케줄러를 사용했고 0.98승씩 메 에포크마다 학습률을 감소시켰다. 학습 조기 종료 제한은 15번의 에포크로 했다. 평가 지표와 손실 함수는

위의 MusicBERT와 동일한 것을 사용했다.

Table 4.4 CoLA 하이퍼파라미터

하이퍼파라미터	값
Num Layers	1
GNN Out Dims	2560
Dropout Rate	0.5
Number of Neighbors	8
Learning Rate	0.0004
Optimizer	Adam
Batch Size	16

모든 모델에 대한 그래프 뉴럴 네트워크의 층이 1개인 이유는 같은 아티스트일 경우 이웃 노드로 정의하였으므로 이웃 노드의 이웃 노드 또한 같은 의미인 같은 아티스트이다. 따라서 이미 타겟 노드의 이웃 노드에 포함되어있기 때문에 이웃 노드의 이웃 노드를 활용할 필요가 없다. 층을 더 쌓는 것이 한 층에서 이웃 노드의 수를 늘리는 것과 의미상 같으므로 1개의 층만 사용했다.

4.3 실험 결과

이번 장에서는 실험 결과에 관해 서술한다. Representation 모델별 실험 결과를 제시하고 결과를 해석한다.

4.3.1 MusicBERT

Dataset	Lakh MIDI	
사용 정보	Method	Micro F1 Score
곡	melody2vec[29]	64.90
	PiRhDy[36]	66.80
	MusicBERT[65]	76.1
아티스트	decision tree	81.38
곡, 아티스트	MusicBERT + artist emb.	81.56
	MusicBERT + GNN	**83.18**
	MusicBERT + DropEdge	82.56

Table 4.5 MusicBERT 실험 결과

MusicBERT의 실험 결과를 표 4.5에 나타냈다. 표에서 곡, 아티스트 정보를 모두 사용하는 행의 artist emb.은 아티스트 정보를 직접적으로 사용한 경우를 뜻한다. 곡만의 정보를 사용할 때보다 아티스트 정보만을 사용할 때 더 높은 성능을 보인다. 또한, 곡의 정보에 추가로 아티스트 정보를 직접적, 간접적으로 사용할 때 성능 향상을 보인다. 직접적으로 사용할 때 7.17%, 간접적으로 사용할 때 9.06%의 향상률을 보인다. 아티스트의 정보를 간접적으로 사용할 경우 직접적 사용보다 높은 성능 향상이 확인된다.

간접적으로 사용할 때는 DropEdge를 적용하지 않은 것이 더 높은 성능을 보인다. 이는 neighbor sampler를 통해 사용하는 이웃 노드의 수를 5개로 줄인 상태에서 DropEdge를 통해 더 줄여 정보를 집계하는 데 사용하는 이웃 노드가 오히려 부족해진 것으로 생각된다.

4.3.2 피아노롤

Dataset	Lakh Pianoroll	
사용 정보	Method	Micro F1 Score
곡	피아노롤[21]	63.70
아티스트	decision tree	80.16
곡, 아티스트	피아노롤 + artist emb.	**81.68**
	피아노롤 + GNN	81.66
	피아노롤 + DropEdge	81.29

Table 4.6 피아노롤 모델 실험 결과

피아노롤 모델의 실험 결과를 표 4.6에 나타냈다. 곡만의 정보를 사용할 때보다 아티스트 정보만을 사용할 때 더 높은 성능을 보이고 곡의 정보에 추가로 아티스트 정보를 직접적, 간접적으로 사용할 때 성능 향상을 보인다. 직접적으로 사용할 때와 간접적으로 사용할 때 모두 28.24%의 성능 향상을 보인다. 아티스트를 직접적으로 사용할 경우 간접적 사용보다 더 높은 성능 향상을 보이지만 그 차이는 매우 미미하다. DropEdge의 성능이 더 낮은 것은 앞서 MusicBERT에서의 실험과 동일한 이유라고 생각된다.

4.3.3 Mixture of CNN

Dataset	FMA Small	
사용 정보	Method	Micro F1 Score
곡	Representation learning using artist labels[45]	57.90
	MuLaP[39]	61.10
	Mixture of CNN[64]	55.88
아티스트	decision tree	
곡, 아티스트	Mixture of CNN + artist emb.	
	Mixture of CNN + GNN	60.62
	Mixture of CNN + DropEdge	**63.00**

Table 4.7 Mixture of CNN 실험 결과

Mixture of CNN의 실험 결과를 표 4.7에 나타냈다. 해당 실험의 사용 데이터셋인 FMA Small은 학습, 검증, 테스트 데이터셋이 나뉘어 미리 정의돼 있다. 이렇게 나눠 정의된 데이터셋에서 아티스트 또한 대부분 나뉘어 있어 테스트 데이터셋에 존재하는 아티스트가 학습 데이터셋에는 존재하지 않은 경우가 많아 아티스트 정보만을 사용한 방법은 사용할 수 없고 아티스트 정보를 사용하기 위해 아티스트 정보를 학습해야 하는 아티스트 직접적 사용 방법은 실험이 무의미하다. 따라서 해당 표 4.7에서는 두 실험을 빈칸으로 나타냈다. 곡의 정보만을 사용한 경우에 비해 곡의 정보에 아티스트 정보를 간접적으로 사용할 때 12.74%의 성능 향상을 보인다. 표 4.7에 나타낸 MuLaP은 FMA Small 데이터셋의 SOTA(state-of-the-art) 모델이다. 본 연구에서 제안한 그래프 뉴럴

네트워크를 통해 아티스트 정보를 간접적으로 추가 사용했을 때 해당 SOTA 모델보다 더 높은 성능을 보인다. 해당 실험에서는 DropEdge를 적용한 실험이 더 높은 성능을 보이는데 이는 해당 데이터셋이 아티스트 당 평균 3.5개의 곡이 존재하므로 다른 데이터셋에 비해 이웃 노드의 수가 적어 다른 데이터셋에 비해 neighbor sampler를 통해 줄어드는 이웃 노드의 수가 노드 별로 적거나 없다. 따라서 정보 집계에 사용하는 이웃 노드의 수가 앞선 모델에서처럼 줄여진 상태에서 DropEdge를 통해 한 번 더 줄인 것이 아니라 기존의 모든 이웃 노드에서 DropEdge를 적용한 것과 같으므로 모델의 강건성을 높이는 DropEdge의 강점이 발휘됐다고 볼 수 있다.

4.3.4 CoLA

Dataset	MGT-Jamendo	
사용 정보	Method	Micro F1 Score
곡	CoLA[52]	39.75
아티스트	decision tree	43.99
곡, 아티스트	CoLA + artist emb.	47.22
	CoLA + GNN	**50.99**
	CoLA + DropEdge	49.51

Table 4.8 CoLA 실험 결과

CoLA의 실험 결과를 표 4.8에 나타냈다. 곡만의 정보를 사용할 때보다 아티스트 정보만을 사용할 때보다 높은 성능을 보이고 곡의 정보에 추가로 아티스트 정보를 직접적, 간접적으로 사용할 때 성능 향상을 보인다. 직접적으로 사용할 때 18.79%, 간접

적으로 사용할 때 22.42%의 성능 향상을 보인다. 아티스트를 간접적으로 사용할 경우 직접적 사용보다 더 높은 성능 향상을 보인다. DropEdge의 성능이 더 낮은 것은 앞선 MusicBERT에서의 실험과 동일한 이유라고 생각된다.

4.4 Label homophily ratio

Dataset	Homophily Ratio
Lakh MIDI	71.10
Lakh Pianoroll	75.02
FMA Small	94.07
MGT-Jamendo	27.39

Table 4.9 각 데이터셋 별 Label Homophily Ratio

Label homophily ratio는 그래프에서 간선으로 연결된 노드끼리 라벨이 일치하는 비율을 뜻한다. Label homophily ratio가 높을수록 그래프에서 집계되는 정보들끼리 상관관계가 높다는 의미이므로 같은 데이터셋 내에서 label homophily ratio가 높을수록 높은 성능을 얻을 확률이 커진다[66]. 하지만 낮은 homophily ratio에 대해서도 그래프 뉴럴 네트워크가 높은 품질의 representation을 학습한다는 연구[38]도 존재한다. 이렇듯 현재 homophily ratio에 대한 논쟁은 계속 이어지고 있다. 따라서 본 연구에서 사용한 데이터셋들의 label homophily ratio를 확인한 후 제안 기법이 다양한 homophily ratio에 어떠한 결과를 가지는지 살펴보고자 한다. 본 연구에서 사용한 네 가지 데이터셋에 대한 label homophily ratio 값은 표 4.9와 같다. 본 연구에서 사용한 데이터셋들은 다양한 label homophily ratio를 가지고 있고 이러한 다양한 값에 대해서 제안 기법이 모두 유효한 성능 향상을 보였다고 할 수 있다.

4.5 아티스트 정보 사용 방법 비교

본 연구에서는 각각의 아티스트에 따라 아티스트 정보를 임베딩을 통해 학습시켜 장르 분류기 앞 단에서 곡의 정보와 concatenation하는 직접적인 방법과 아티스트 정보를 그래프 구조에서 간선을 정의하는 용도로 사용하는 간접적인 방법, 총 두 가지 방법을 사용하였다. 피아노롤 모델에서의 미미한 차이를 제외하고 모든 모델에서 아티스트 정보를 직접적으로 활용하는 방법보다 간접적으로 활용하는 방법의 성능이 높았다. 또한, 4.3.3절에서 아티스트 정보 직접적 사용의 한계를 확인할 수 있었다. FMA 데이터셋과 같이 학습 데이터셋과 테스트 데이터셋에 같은 아티스트가 존재하지 않으면 해당 아티스트에 대한 정보를 학습할 수 없고 학습하더라도 테스트 데이터셋에 사용하지 못하므로 무의미하다. 이에 비해 아티스트 정보 간접적 사용은 단순히 노드인 곡의 정보끼리 간선으로 연결해주는 역할을 하므로 아티스트에 대한 임베딩을 따로 학습할 필요가 없다. 따라서 학습 데이터셋과 테스트 데이터셋에 동일한 아티스트가 존재하지 않아도 된다. 만약, 한 아티스트에 대해 하나의 곡만 존재하여 이웃 노드가 정의되지 않더라도 노드는 고립된 노드로서 집계할 이웃 노드가 없을 뿐 자신의 곡 정보만을 사용해 장르를 분류하면 되기 때문에 해당 제안 기법을 적용할 수 있다.

아티스트 정보를 직접적으로 사용할 때의 또 다른 한계로 간접적 사용보다 강건성이 낮다는 것이다. 각 아티스트 임베딩은 모든 데이터에 대해 학습되는 것이 아니라 각각의 동일한 아티스트의 데이터에 대해서만 학습된다. 이에 비해 아티스트 간접적 사용은 모든 데이터에 대해 계산 그래프의 파라미터가 학습되고 공유되므로 아티스트 정보가 각각 아티스트 저마다의 임베딩을 통해 학습되는 직접적 사용보다 모든 데이터를 통해 계산 그래프의 파라미터가 학습되는 간접적 사용이 강건성이 높다.

4.6 장르 개수에 따른 베이스라인 모델과의 성능 차이 및 각 모델별 성능 표준 편차

소수의 특정 장르만을 가진 아티스트가 대다수이지만 여러 가지 장르를 가지는 아티스트도 존재한다. 예를 들어, 한국의 유명 가수 조용필은 록, 발라드, 트로트, 전자음악, 오페라, 퓨전 재즈와 같이 다양한 장르를 가진다. 아티스트가 여러 가지 장르를 가지는 위 같은 경우 아티스트 정보를 간접적으로 사용할 때 연결된 노드들의 장르가 다르므로 이웃 노드들의 정보가 노이즈로 작용할 가능성이 존재한다. 따라서 아티스트가 가지는 장르의 수에 따른 곡만을 사용한 모델과의 성능 차이를 비교한다. 또한, 장르 수에 따른 성능의 표준 편차를 관찰한다. FMA Small 데이터셋은 아티스트 별 장르 수가 대다수 일정하므로 해당 데이터셋을 통해 학습한 Mixture of CNN을 제외한 세 가지 모델들에 관한 결과를 관찰한다.

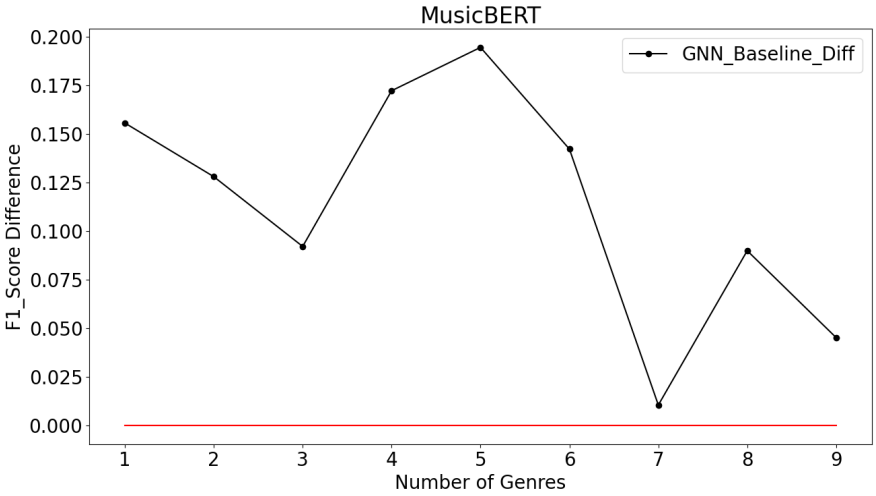

Figure 4.5 MusicBERT - 장르 개수 별 베이스라인과 제안 기법 성능 차이

첫 번째로 MusicBERT의 장르 개수에 따른 성능에 관해 서술한다. 그림 4.5의 X축은 장르의 수이고 Y축은 곡만을 사용한 방법과 아티스트 정보 간접적 사용 방법의

	Baseline	GNN	DropEdge
표준 편차	0.1478	0.1649	0.1528

Table 4.10 MusicBERT - 장르 개수에 따른 성능 표준 편차

성능 차이다. 장르의 수가 늘어남에 따라 곡만 사용한 모델과의 차이가 대체로 점점 줄어드는 것을 알 수 있다. 하지만 여전히 곡만을 사용한 모델보다는 높은 성능을 보이고 있다. 표 4.10를 보면 장르 개수에 따른 성능 표준 편차가 나타나 있다. 곡만을 사용한 모델이 장르 개수에 따라 가장 적은 표준 편차를 가지고 있고 그래프 뉴럴 네트워크에 DropEdge를 적용한 모델, 적용하지 않은 모델 순으로 표준 편차가 증가한다. 따라서 DropEdge를 적용할 경우 장르 수 관점에서는 모델이 더 강건해진다는 것을 알 수 있고 아티스트 정보를 사용하면 모델이 비교적 불안정해지는 것을 알 수 있다. 하지만 많은 장르를 가지고 있더라도 곡만을 사용한 방법보다는 여전히 높은 성능을 유지하고 있다.

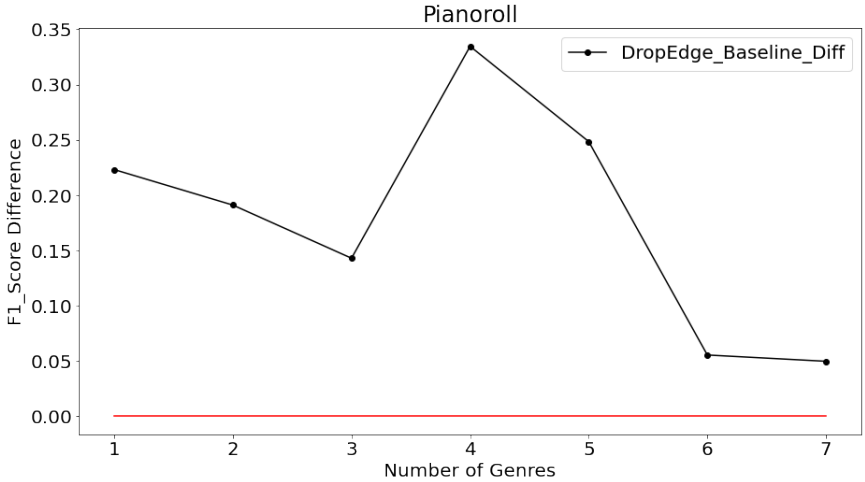

Figure 4.6 피아노롤 - 장르 개수 별 베이스라인과 제안 기법 성능 차이

다음으로 피아노롤 모델의 장르 개수에 따른 성능을 관찰한다. 그림 4.6을 보면

	Baseline	GNN	DropEdge
표준 편차	0.1145	0.1332	0.1213

Table 4.11 피아노롤 - 장르 개수에 따른 성능 표준 편차

MusicBERT의 경우와 동일하게 장르 개수가 증가함에 따라 곡만을 사용한 모델과의 성능 차이가 줄어들고 있다. 또한, 표 4.11를 통해 곡만을 사용한 방법보다 아티스트 정보를 추가하였을 때 더 높은 표준편차를 보이는 것도 관찰된다. 피아노롤 모델에서도 앞선 MusicBERT와 마찬가지로 장르의 수가 증가함에 따라 곡만을 사용한 모델과의 성능 차이는 점점 줄어들고 곡만을 사용한 모델보다 표준 편차도 높지만, 여전히 더 높은 성능을 보이고 있다.

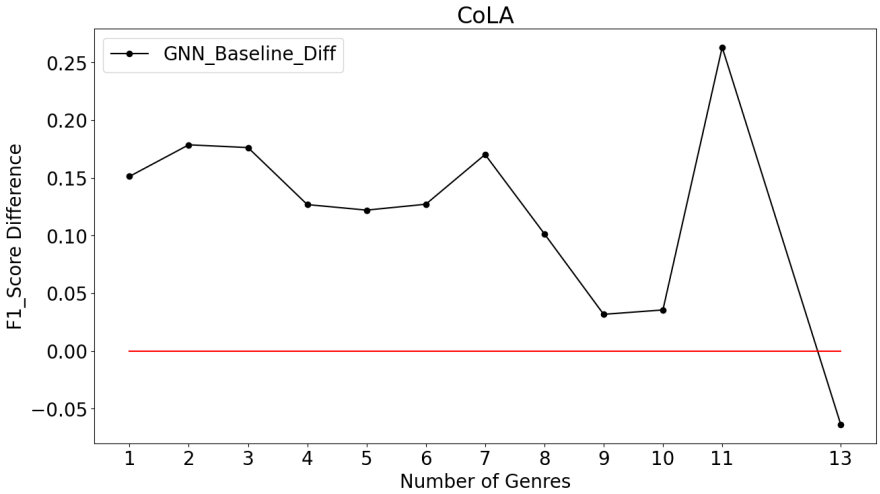

Figure 4.7 CoLA - 장르 개수 별 베이스라인과 제안 기법 성능 차이

	Baseline	GNN	DropEdge
표준 편차	0.0616	0.1227	0.1295

Table 4.12 CoLA - 장르 개수에 따른 성능 표준 편차

마지막으로 CoLA의 장르 개수에 따른 성능을 관찰한다. 앞선 두 모델과 마찬가지로

장르의 개수가 증가함에 따라 곡만을 사용한 방법과의 성능 차이가 점점 줄어들고 있다. 그러다 장르 개수가 13개가 됐을 때엔 곡만을 사용한 방법의 성능이 아티스트 정보를 활용한 방법보다 0.06만큼 더 높은 성능을 보인다. 이때, 이러한 13개의 장르를 가지는 아티스트의 수는 5명이고 해당 아티스트에 대한 데이터의 수는 33개이다. 이는 총 테스트 데이터 4,864개 중 0.6%의 비율이고 33개의 데이터 중 1개 정도의 데이터에 대한 성능 차이로 생각된다. 하지만 장르의 개수가 더욱 많아질 경우 해당 문제는 더 심화될 수 있다. 따라서 해당 결과는 본 제안 기법의 한계라고 판단되고 향후 연구(5.3절)에서 한계를 완화하거나 해결할 수 있는 방법들을 제안한다.

제 5 장 결론

5.1 결론

본 논문에서는 아티스트 정보를 효과적으로 활용할 수 있는 그래프 뉴럴 네트워크 기반 플러그 앤 플레이 프레임워크를 제안하였다. 플러그 앤 플레이 프레임을 검증하기 위해 네 가지 오픈 데이터셋에 적용하였고 네 가지 representation 모델을 활용하였다. 해당 기법의 성능을 평가하기 위해 곡만을 활용한 방법, 아티스트만을 활용한 방법, 직접적으로 아티스트 정보를 활용한 방법과 비교 실험하였다. 피아노롤 모델에서의 미미한 차이를 제외하고 모든 실험에 대해서 제안 기법이 가장 높은 성능을 보임을 확인했다. 아티스트 정보를 직접적으로 활용한 방법과 제안 기법을 비교하여 제안 기법의 상대적 강점을 보였다. 모든 실험 데이터셋에 대한 label homophily ratio를 제시함으로 다양한 label homophily ratio에 대해서 제안 기법이 효과적이라는 것을 확인하였다. 아티스트 정보를 간접적으로 활용할 때 아티스트가 여러 가지 장르를 가진다면 이웃 노드의 정보를 집계함에 있어 해당 노드들의 정보가 노이즈로 작용할 가능성이 존재하므로 곡만을 활용한 방법과 제안 기법을 장르 수에 따라 성능을 비교하였다. 장르 수가 증가할수록 곡만을 활용한 방법과의 성능 차이는 점점 작아지지만, 장르의 수가 13개일 경우를 제외하고는 아티스트 정보를 그래프 뉴럴 네트워크를 통해 간접적으로 활용한 본 논문의 제안 기법이 더 높은 성능을 보였다.

5.2 한계

4.6절에서 볼 수 있듯이 한 아티스트가 매우 많은 장르를 가질 때 본 논문의 제안 기법은 곡만을 사용한 모델보다 더 낮은 성능을 보인다. 또한, 본 연구에서 사용한 모든 데이터셋들은 여러 가지 장르를 가지지 않는 아티스트가 대다수이므로 여러 가지 장르를 가지는 아티스트가 대다수일 경우 성능 향상을 보장하지 못한다. 아티스트와 장르와의 상관관계가 매우 낮을 경우, 즉, 아티스트가 극단적으로 곡마다 새로운 장르를 생성할 경우 본 연구의 제안 기법의 성능을 보장하기 힘들다. 본 연구에서 총 9만여 개의 곡으로 실험을 진행하였고 label homophily ratio가 27.39인 MGT-Jamendo 데이터셋에 대한 실험에서도 성능이 향상됐지만 아티스트 정보를 활용함에 있어 근본적인 한계를 벗어나기에는 추가적인 기법이 필요해 보인다.

5.3 향후 연구

본 연구에서는 아티스트 정보를 간접적으로 활용하여 그래프 뉴럴 네트워크 구조를 사용했다. 5.2절에서 언급했듯이 이러한 아티스트 정보를 활용하는 데에는 근본적인 한계가 존재하고 이를 해결하기 위해서는 추가 기법이 더 필요하다. 같은 아티스트일 때 노드끼리 간선으로 연결하는 거에 더하여 연결되는 노드끼리의 representation 유사도를 계산해 유사도가 특정 기준보다 낮을 때 간선으로 연결하지 않는 방법에 관해서 추후 연구를 진행할 수 있다. 또한 아티스트 정보 사용 없이 단지 representation끼리의 유사도를 통해 이웃 노드를 정의할 수 있다. 하지만 아티스트 정보를 사용함으로써 계산 복잡도를 매우 줄일 수 있다. 데이터의 수가 늘어남에 따라 모든 데이터 쌍끼리의 유사도를 계산하여야 하므로 제곱의 연산량이 필요하다. 하지만 같은 아티스트에 대해서만 유사도를 계산할 경우 계산량을 많이 감소시킬 수 있다. 다른 방법으로는 representation을 클러스터링하여 클러스터에 속하지 않는 노드들의 간선을 끊는 방법도 있다. 또한

아티스트 정보가 아닌 곡의 bpm과 같은 다른 사이드 정보를 활용하는 방법도 추후 연구할 수 있다. 본 연구에서는 그래프 뉴럴 네트워크 모델을 기본적인 GraphSAGE를 사용했지만 추후 연구에서는 더욱 고도화되고 강건한 모델을 사용할 수 있다.

참고 문헌

[1] *Graph Neural Network Structure*, Stanford University. `http://web.stanford.edu/class/cs224w`.

[2] *MIDI vs. Digital Audio*, Rajdhani College. `https://rajdhanicollege.ac.in/admin/ckeditor/ckfinder/userfiles/files/BAprogIII_28Mar_2T.pdf`.

[3] 그래프 뉴럴 네트워크 구조, Naver BoostCamp AI Tech. `https://boostcamp.connect.or.kr`.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[5] J. Alammar, *The illustrated bert, elmo, and co. (how nlp cracked transfer learning)*, 2018.

[6] P. Alonso-Jiménez, X. Serra, and D. Bogdanov, *Music representation learning based on editorial metadata from discogs*, (2022).

[7] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., *Relational inductive biases, deep learning, and graph networks*, arXiv preprint arXiv:1806.01261, (2018).

[8] D. Beck, G. Haffari, and T. Cohn, *Graph-to-sequence learning using gated graph neural networks*, arXiv preprint arXiv:1806.09835, (2018).

[9] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, *The mtg-jamendo dataset for automatic music tagging*, (2019).

[10] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, *End-to-end incremental learning*, in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 233–248.

[11] Z. Cataltepe, Y. Yaslan, and A. Sonmez, *Music genre classification using midi and audio features*, EURASIP Journal on Advances in Signal Processing, 2007 (2007), pp. 1–8.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, in International conference on machine learning, PMLR, 2020, pp. 1597–1607.

[13] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, Y.-H. Chen, S.-W. Li, and H.-y. Lee, *Audio albert: A lite bert for self-supervised learning of audio*

representation, in 2021 IEEE Spoken Language Technology Workshop (SLT), IEEE, 2021, pp. 344–350.

[14] Y.-H. Chou, I. Chen, C.-J. Chang, J. Ching, Y.-H. Yang, et al., *Midibert-piano: large-scale pre-training for symbolic music understanding*, arXiv preprint arXiv:2107.05223, (2021).

[15] C. Cortes and V. Vapnik, *Support-vector networks*, Machine learning, 20 (1995), pp. 273–297.

[16] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, *Fma: A dataset for music analysis*, arXiv preprint arXiv:1612.01840, (2016).

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[18] S. Dixon, F. Gouyon, G. Widmer, et al., *Towards characterisation of music via rhythmic patterns.*, in ISMIR, 2004.

[19] K. Do, T. Tran, and S. Venkatesh, *Graph transformation policy network for chemical reaction prediction*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 750–760.

[20] S. Dokania and V. Singh, *Graph representation learning for audio & music genre classification*, arXiv preprint arXiv:1910.11117, (2019).

[21] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, *Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[22] M. Dong, *Convolutional neural network achieves human-level accuracy in music genre classification*, arXiv preprint arXiv:1802.09697, (2018).

[23] W. J. Dowling and D. L. Harwood, *Music cognition*, Academic press, 1986.

[24] X. Favory, K. Drossos, T. Virtanen, and X. Serra, *Coala: Co-aligned autoencoders for learning semantically enriched audio representations*, arXiv preprint arXiv:2006.08386, (2020).

[25] M. Fey and J. E. Lenssen, *Fast graph representation learning with PyTorch Geometric*, in ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[26] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, *Protein interface prediction using graph convolutional networks*, Advances in neural information processing systems, 30 (2017).

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[28] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, Advances in neural information processing systems, 30 (2017).

[29] T. Hirai and S. Sawada, *Melody2vec: Distributed representations of melodic phrases based on melody segmentation*, Journal of Information Processing, 27 (2019), pp. 278–286.

[30] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[32] M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi, *The similarity metric*, IEEE transactions on Information Theory, 50 (2004), pp. 3250–3264.

[33] R. Li, M. Tapaswi, R. Liao, J. Jia, R. Urtasun, and S. Fidler, *Situation recognition with graph neural networks*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 4173–4182.

[34] T. Li, M. Ogihara, and Q. Li, *A comparative study on content-based music genre classification*, in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003, pp. 282–289.

[35] T. L. Li and A. B. Chan, *Genre classification and the invariance of mfcc features to key and tempo*, in International Conference on MultiMedia Modeling, Springer, 2011, pp. 317–327.

[36] H. Liang, W. Lei, P. Y. Chan, Z. Yang, M. Sun, and T.-S. Chua, *Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music*, in Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 574–582.

[37] B. Logan, *Mel frequency cepstral coefficients for music modeling*, in In International Symposium on Music Information Retrieval, Citeseer, 2000.

[38] Y. Ma, X. Liu, N. Shah, and J. Tang, *Is homophily a necessity for graph neural networks?*, in International Conference on Learning Representations, 2021.

[39] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, *Learning music audio representations via weak language supervision*, in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 456–460.

[40] P. Manocha, Z. Jin, R. Zhang, and A. Finkelstein, *Cdpam: Contrastive learning for perceptual audio similarity*, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 196–200.

[41] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason,

E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, A. Weiss, D. Hereñú, F.-R. Stöter, L. Nickel, P. Friesch, M. Vollrath, and T. Kim, *librosa/librosa: 0.9.2*, June 2022.

[42] C. McKay, *Automatic genre classification of midi recordings*, (2004).

[43] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 26 (2013).

[44] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, *fairseq: A fast, extensible toolkit for sequence modeling*, in Proceedings of NAACL-HLT 2019: Demonstrations, 2019.

[45] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, *Representation learning of music using artist labels*, arXiv preprint arXiv:1710.06648, (2017).

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[47] L. E. Peterson, *K-nearest neighbor*, Scholarpedia, 4 (2009), p. 1883.

[48] J. R. Quinlan, *Induction of decision trees*, Machine learning, 1 (1986), pp. 81–106.

[49] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*, Columbia University, 2016.

[50] Y. Rong, W. Huang, T. Xu, and J. Huang, *Dropedge: Towards deep graph convolutional networks on node classification*, in International Conference on Learning Representations, 2019.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*, tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[52] A. Saeed, D. Grangier, and N. Zeghidour, *Contrastive learning of general-purpose audio representations*, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 3875–3879.

[53] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model*, IEEE transactions on neural networks, 20 (2008), pp. 61–80.

[54] M. Tan and Q. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, in International conference on machine learning, PMLR, 2019, pp. 6105–6114.

[55] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, *Linear discriminant analysis: A detailed tutorial*, AI communications, 30 (2017), pp. 169–190.

[56] S. Thrun and L. Pratt, *Learning to learn: Introduction and overview*, in Learning to learn, Springer, 1998, pp. 3–17.

[57] G. Tzanetakis and P. Cook, *Musical genre classification of audio signals*, IEEE Transactions on speech and audio processing, 10 (2002), pp. 293–302.

[58] G. Velarde, T. Weyde, C. C. Chacón, D. Meredith, and M. Grachten, *Composer recognition based on 2d-filtered piano-rolls*, in International Society for Music Information Retrieval Conference, International Society for Music Information Retrieval, 2016, pp. 115–121.

[59] L. Wang and A. v. d. Oord, *Multi-format contrastive learning of audio representations*, arXiv preprint arXiv:2103.06508, (2021).

[60] H. Weng, J. Chen, D. Wang, X. Zhang, and D. Yu, *Graph-based attentive sequential model with metadata for music recommendation*, IEEE Access, 10 (2022), pp. 108226–108240.

[61] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, *Simplifying graph convolutional networks*, in International conference on machine learning, PMLR, 2019, pp. 6861–6871.

[62] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, *A comprehensive survey on graph neural networks*, IEEE Transactions on Neural Networks and Learning Systems, 32 (2020), pp. 4–24.

[63] L. Yao, C. Mao, and Y. Luo, *Graph convolutional networks for text classification*, in Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 7370–7377.

[64] Y. Yi, K.-Y. Chen, and H.-Y. Gu, *Mixture of cnn experts from multiple acoustic feature domain for music genre classification*, in 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2019, pp. 1250–1255.

[65] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, *Musicbert: Symbolic music understanding with large-scale pre-training*, in Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021, pp. 791–800.

[66] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, *Interpreting and unifying graph neural networks with an optimization framework*, in Proceedings of the Web Conference 2021, 2021, pp. 1215–1226.

Abstract

Improving the Performance of Music Genre Classification through Exploiting Artist Information

Seokgi Lee
Department of Industrial Engineering
The Graduate School
Seoul National University

Music genre classification is the task of classifying target music into specific genres such as blues, jazz, classical music, pop, etc. In music genre classification, it is critical to identify and extract the characteristics of each genre by using the fact that each genre has different elements of music. Therefore, the deep learning model in which feature extraction and classification processes are designed through a single model has recently attracted much attention. In this paper, a deep learning model is used to classify music genres.

The majority of deep learning models classify music genres using only the information of songs. However, many people predict the genre of the song only through its artists, which shows high accuracy. Focusing on this point, this paper tries to use artist information, which is sides information, for the classification of music

genres through a deep learning model. Confirm through four open datasets that artist information plays a major role in genre classification on four open datasets through genre classification experimental results using only artist information, and propose a graph neural network-based plug-and-play framework that can effectively exploit artist information. This paper compares and experiments with the method of directly providing artist information through artist embedding and the method of indirectly providing it through graph neural networks and presents the limitations of the method of directly providing artist information and the advantages of graph neural networks.

In music genre classification research, deep learning models train musical characteristics by using various data types of music. There are two representative data formats: a type of converting audio data into an image form and a type of converting into a symbolic form. This paper experiments with the proposed framework on open datasets in symbolic and audio forms to verify that the corresponding framework is plug-and-play. In addition, this paper applies the proposed framework to four models that are trained with the corresponding dataset and compares each with the proposed framework results. The four models trained to classify music genres in different ways. Two models were trained with supervised learning, and the other two models were pre-trained with unsupervised learning and then fine-tuned to classify music genres with supervised learning. This paper confirms that the proposed framework is applicable in various environments through models learned in various ways.

Keywords: Music Genre Classification, Artist Information, Graph Neural Network,

Plug&Play

Student Number: 2021-22750