Music Recommendation Using Deep Learning

Alexandros Liapatis
School of Applied Mathematical and Physical Sciences, Dpet. of Physics, NTUA
Athens, Greece
alexandrosliapates@gmail.com

Ioannis Anastasopoulos
School of Information and Communication Technologies, Dpet. of Informatics
University of Piraeus (UNIPI)
Piraeus, Greece
i.anastasopoulos@outlook.com

Evaggelos Anagnostopoulos
University of Ioannina (UOI)
Department of Physics
Athens, Greece
evanganag@gmail.com

***Abstract*—Music recommendation systems are pivotal in enhancing user experience by offering personalized music suggestions tailored to individual preferences. This paper explores a content-based filtering approach for music recommendation, utilizing deep learning models trained on diverse music classification tasks. Our methodology involves training convolutional neural networks (CNNs) to extract embeddings from music audio segments, which are subsequently employed for similarity-based recommendation. We leverage FAISS (Facebook AI Similarity Search) for efficient and precise similarity search in the high-dimensional embedding space. Results demonstrate promising performance in recommending music based on genre, emotion, and musical instrument similarity, underscoring the efficiency of our approach in personalized music discovery. Future directions encompass expanding dataset variety and refining the recommendation engine to cater to a broader range of music preferences and user contexts.**

I. Introduction

A music recommendation system is an application that offers tailored music suggestions to users based on their preferences, listening habits, and previous activity. The advent of digital music streaming platforms and online music stores has resulted in an explosion of available music, making it increasingly challenging for users to discover new songs.

Many recommender systems rely on usage patterns: the combinations of items that users have consumed or rated provide information about the users' preferences, and how the items relate to each other. This is the collaborative filtering approach. Another approach is to predict user preferences from item content and metadata.

The consensus is that collaborative filtering will generally outperform content-based recommendation. However, it is only applicable when usage data is available. Collaborative filtering suffers from the cold start problem: new items that have not been consumed before cannot be recommended. Additionally, items that are only of interest to a niche audience are more difficult to recommend because usage data is scarce. In many domains, and especially in music, they comprise the majority of the available items, because the users' consumption patterns follow a power law. Content-based recommendation is not affected by these issues. For the purposes of this project we focus on the second method, content based filtering.

In the next sections we will discuss about related work, the datasets, the methodology used to tackle content based filtering and the results of the experiment.

II. Related work

Recent advancements in deep learning have led to the development of sophisticated models for music recommendation. These models often employ convolutional neural networks (CNNs) to process spectrograms derived from audio signals. A study by Ryan Whitell [5] demonstrates that musically-motivated CNN architectures, tailored for audio data rather than repurposed from image processing tasks, yield better performance in music recommendation tasks. These models excel in creating embeddings that effectively capture the nuances of musical content, facilitating accurate similarity searches and recommendations.

These CNNs can also be utilized in tasks such as music genre classification, mood detection, and instrument classification, which can be used to craft more refined and personalized recommendation systems. The ability to understand and preserve the intricate details of audio signals allows CNN-based models to analyze the actual sound waves, thereby capturing a more authentic representation of the music.

Whitell's study highlights several innovative architectural choices that enhance the performance of CNNs in music recommendation systems. These include the use of specialized filter shapes and sizes that align with the properties of musical signals, as well as the incorporation of temporal and spectral features that are unique to audio data. By focusing on these aspects, the CNNs can generate high-quality embeddings that reflect the true characteristics of the music, leading to more precise and personalized recommendations for users.

In conclusion, the adoption of CNN architectures tailored specifically for audio signals represents a significant leap forward in the field of music recommendation. As research and technology continue to evolve, we can expect even more sophisticated and user-centric music recommendation systems, further enriching the way we discover and enjoy music.

III. Dataset

A. GTZAN Dataset

The GTZAN dataset [4] is a well-known dataset for musical genre classification of audio signals. It consists of 1,000 audio tracks, each 30 seconds long, covering 10 genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. Each genre is represented by 100 tracks. The audio tracks are in WAV format Mono 16-bit audio. For our project, we split each track into 10 second segments, in order to perform genre classification.

B. Musical Instrument's Sound Dataset

The Musical Instrument's Sound Dataset is a curated collection featuring audio recordings from four distinct musical instrument classes: Guitar, Drum, Violin, and Piano. Initially, each class contained 700 sounds in the training set, except for the Piano, which had 528 sounds. To maintain consistency, we kept all the audio files that were at least 10 seconds long. Then we cropped the remaining files to be exactly 10 seconds. Finally, we concluded to a balanced dataset with approximately 100 instances per class.

C. Emotion Classification Dataset

The MIREX-like Mood Dataset [3] is resource for emotion classification in music. It comprises 903 30-second audio clips, organized into five clusters and subcategories representing different mood labels. Additionally, it includes 764 lyric files in text format and 196 MIDI files. For our project, we focused exclusively on the audio components, converting the MP3 files to WAV mono format for consistency. We then cropped each song to 10 seconds segments and balanced the classes to approximately 500 instances per class.

D. FMA: A Dataset For Music Analysis

The FMA dataset [1] format is very similar to the GTZAN format. It consists of 8000 30 seconds tracks. We proceeded with the small version where there are 8 classes, Rock, Pop, Electronic, Experimental, Folk, Hip-Hop, Instrumental, International. They come in two channel MP3 format. The files were converted to WAV mono channel format. The dataset came along with some metadata containing information like artists, bit-rate, composer, etc. The metadata was stored in a SQL database to map each audio file to each corresponding metadata features.

IV. Methodology

The aforementioned four datasets were used for the purposes of this project, three of them were used to build models trained on different downstream classification tasks (e.g. emotion, instrument, genre recognition) and the fourth one, FMA, was kept unseen to test the recommendation capabilities afterwards.

A. Model

As far as the model is concerned, the same configuration was used for every training task. The model was a simple CNN with four convolutional layers followed by three linear layers where the layer before the classifier has an output size of 256. Audio signals that were fed into the model were sampled with a 50 millisecond window length and an equally long hop length. So there was no overlap between sample windows.

B. Embedding extraction

During training, three identical CNN models were created, each one trained on a different classification task. While the performance on each task is not that important for the purpose of the project, it is necessary that the model can adequately discriminate between the classes of its respective task. The better the performance of the model, the more informative the embeddings of an audio signal will be with respect to its class.

All of our models have been trained on 10 second segment audio signals so we do the same for the FMA files, resulting in 3 segments per file. Every segment passes through each one of the models to get the embeddings. So from every model we get three different embeddings per track. If we multiply this with the number of models we get 9 different embeddings per track. In order to not lose any information from the representations, we just concatenate all of them in the end, getting an embedding of $9 \times D$, where D is the embedding dimension. As mentioned before, in the Model section, the embedding dimension is 256, so we get an embedding of size 2.304 per file.

C. Similarity Search

In order to find the most similar embeddings given a search query, we need to implement a vector searching mechanism. Since we are dealing with high-dimensional vectors, we need to index those vectors for fast and approximate nearest neighbor search (ANN). When given a search query, which is also represented as a high-dimensional vector, our similarity search mechanism must find quickly the vectors, in the indexed vector space, that are most similar to the query vector. This similarity is often measured using distance metrics in the embedding space where our vectors are represented. These metrics consist of, but are not limited to, the Euclidean distance and cosine similarity.

Those metrics are computed as follows:

The Euclidean distance between two vectors a and b in an n-dimensional space is given by:

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{1}$$

While the cosine similarity between two vectors a and b is given by:

$$cosine\ similarity(a,b) = \frac{a \cdot b}{\|a\|\|b\|}$$

$$= \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \quad (2)$$

In order to achieve this similarity search in an efficient way, we need to utilize a vector database mechanism which will map indexes like the before mentioned ones in our embeddings. The library which we decided to use it FAISS (Facebook AI Similarity Search) [2]. This similarity engine operates both as a vector database in our use case, as well as a similarity search mechanism. It uses various algorithms and data structures in its search mechanism, such as inverted file systems, clustering (similar to k-means) and quantization techniques, which significantly reduce the search space and computational complexity, since most indexes are loaded in memory, allowing for rapid and scalable searches even in large datasets. We have narrowed down the choice of the index since FAISS offers a variety of search indexes each specialised for a specific search case.

In our approach, we opted to use the "FlatL2" and "IVFFlat" indexes provided by the FAISS library for efficient similarity search across our high-dimensional embeddings. This decision is grounded in the specific requirements of our use case and the characteristics of these indexing methods.

1) FlatL2 Index: The "FlatL2" index in FAISS is a straightforward, brute-force method that computes the exact Euclidean distance between vectors. Each query vector is compared against all vectors in the dataset, ensuring that the most similar vectors are accurately identified. This method is advantageous in scenarios where exactness is paramount, particularly when dealing with high-dimensional data like our 2304-dimensional embeddings. Given the high precision needed for downstream tasks, the "FlatL2" index ensures that we do not compromise on the accuracy of our similarity search results.

Despite its computational intensity, the "FlatL2" index is manageable within our computational resources, and it provides a reliable baseline for performance evaluation.

2) IVFFlat Index: The "IVFFlat" (Inverted File with Flat Quantization) index is designed to enhance search efficiency without significantly sacrificing accuracy. It achieves this by partitioning the dataset into clusters using k-means clustering. Each cluster has a centroid, and during the search, only the clusters closest to the query vector are considered. This significantly reduces the number of distance computations required.

The process begins with a training phase where the dataset is divided into k clusters, each represented by a centroid. For a query vector $\mathbf{q}$, the search process involves:

- Identifying the nearest centroids to $\mathbf{q}$.
- Performing a brute-force search within these selected clusters.

This two-step approach balances the trade-off between search speed and accuracy. The "IVFFlat" index is particularly suitable for large-scale datasets like ours, where a purely brute-force approach could be computationally prohibitive. This also offers a scalable solution in case our vectors in our database increase in size over time.

Given the high dimensionality of our embeddings and the need for efficient search mechanisms, the "IVFFlat" index provides a practical solution by reducing search time while maintaining a reasonable level of accuracy. This index leverages the structure in the data to accelerate the search process, providing a feasible solution in case our use cases differ in the future, or we enhance our recommendation engine with additional data.

D. Pipeline

In order to conclude this section, the pipeline followed needs to be clarified. Firstly, three CNN models were trained on separate classification tasks, emotion, instrument and genre classification. Secondly, those three models were used to create the embeddings for the FMA tracks. Three segments per file were passed through each one of the models, resulting in 9 different representations, which were concatenated afterwards. Thirdly, we indexed those 2304 dimensional representations using the FAISS IVFFlat and FlatL2 indexers.

After this procedure, the system is ready to handle queries. The pipeline followed here is quite similar. Given an audio signal of 30 seconds, we split it to 10 second segments and pass them through every model to get our 2304 dimensional embedding. One of the indexers is then queried to search for the model similar embeddings. The index returns the most similar vector indices and their distances from the query vector. Using these vector indices, we can then search our database which includes the metadata information to fetch track details. In the end, what we end up with is a list of the five most similar tracks from our FMA database.

V. Results

Table 1 presents the performance of our three CNN models trained for genre, emotion, and instrument classification. While performance is not our primary focus, it is included here to ensure a comprehensive discussion.

TABLE I
CNN Model Performacne per Task

Classification Task	Accuracy	F1	CE Loss
Genre	0.668	0.674	0.029
Emotion	0.464	0.440	1.307
Instrument	0.810	0.802	0.895

Systems like these can only be tested empirically and that is what we do in the following section.

VI. Evaluation

Tables 2 and 3 illustrate the system's recommendations when the same Rock song is used as a query for comparison. The distinction between the two tables lies in the embeddings employed: Table 2 utilizes embeddings from only the genre and instrument models, while Table 3 employs embeddings from all models. Clearly, the latter approach yields superior performance due to the greater amount of information captured by the embeddings. For both cases, the FlatL2 indexer was utilized.

TABLE II
Similarity Search
with genre & instrument embendings

Recommendations			
Track ID	Title	Artist	Genre
21400	Un Don D'ovules Porteur De Vie	Montag	Pop
40986	Doubt	The Corin Tucker Band	Rock
73520	Pride	Hounds Of Hate	Rock
126668	I'll Take Care Of It	Sleep Out	Rock
126669	Lake On Draft	Sleep Out	Rock
127299	With the Summer	Oh Yeah, the Future	Pop

TABLE III
Similarity Search with all the embendings

Recommendations			
Track ID	Title	Artist	Genre
40986	Doubt	The Corin Tucker Band	Rock
73520	Pride	Hounds Of Hate	Rock
114415	Death Of An Orchid	Scott Holmes	Rock
126668	I'll Take Care Of It	Sleep Out	Rock
126669	Lake On Draft	Sleep Out	Rock
127299	With the Summer	Oh Yeah, the Future	Pop

This example demonstrates the capabilities of our system, though it occasionally recommended songs that were quite dissimilar to the queried one. This variance is likely attributable to the audio quality of the test dataset. Additionally, for simplicity, we converted the audio from stereo to mono channel format, which may have further reduced the audio quality. Despite these challenges, the results showcase the system's potential in diverse conditions.

VII. Conclusions

Music recommendation is a hard task to solve. Finding good quality datasets with the necessary licenses is a difficult task. As a result of our experiment we found that using embeddings from models that have been trained on various different tasks related to audio, even though their performance on the respective task is not perfect, can perform very well on recommendation setting. All the source code can be found at Songs Recommender.

VIII. Future work

In this study, we have demonstrated the effectiveness of musically-motivated CNN architectures. Additionally, we implemented a practical similarity search mechanism using the embeddings generated by these CNNs. By utilizing FAISS (Facebook AI Similarity Search), we efficiently performed similarity searches within a high-dimensional embedding space, highlighting the potential of our approach for personalized music discovery.

However, there is significant potential for further enhancement and sophistication in our methodology. One promising direction for future research is the incorporation of metadata into the embeddings generated by our CNN models. Metadata such as artist information, album details, genre tags, release dates and even favorite count, can provide valuable contextual information that may help in refining the recommendations. By encoding this metadata within the embeddings, we can develop a more comprehensive model that considers both the audio content and the contextual attributes of the music. Additionally, leveraging user interaction data such as likes, dislikes, playlist additions, skips, and listening history can significantly enhance the personalization of the recommendation system. User behavior data provides direct feedback on individual preferences, allowing the system to adapt and optimize recommendations dynamically. Incorporating these interaction metrics can help in building a more adaptive system that continuously learns from user feedback and improves its recommendations over time.

Exploring more complex neural network architectures that can handle combined inputs of audio features, metadata, and user interaction data is a promising direction. Models such as recurrent neural networks (RNNs) or transformers can be used to capture temporal dependencies in user behavior and music consumption patterns. This will enable the system to make more accurate predictions and provide recommendations that align with the user's evolving tastes. Increasing the variety and diversity of datasets used for training and testing the models can improve the robustness and generalization of the recommendation system. Using datasets with a wider range of genres, instruments, moods, and cultural backgrounds will help the system meet a broader range of music preferences and user demographics.

This aligns with approaches used in existing systems such as Spotify, which leverages a combination of content-based and collaborative filtering methods to refine their music recommendation algorithms and enhance user satisfaction. [5]

By pursuing these future research directions, we aim to create a more sophisticated and user-centric music recommendation system that not only understands the nuances of

musical content but also adapts to the unique preferences and behaviors of individual users. This will further enrich the user experience, making music discovery more intuitive and enjoyable.

REFERENCES

[1] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[2] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[3] Renato Panda, Ricardo Malheiro, Bruno Rocha, António Oliveira, and Rui Pedro Paiva. Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. 10 2013.

[4] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals, 2001.

[5] Ryan Whitell. Content-based music recommendation using deep learning, 2019. Regis University Student Publications (comprehensive collection). 968.